MASTER EN SCIENCES INFORMATIQUES

L'analyse cladistique : problème et solutions heuristiques informatisées

d'Udekem d'Acoz Gevers, Marie

Award date:
1989

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE - DAME DE LA PAIX

NAMUR

Institut d'Informatique

L ' ANALYSE CLADISTIQUE : PROBLEME ET SOLUTIONS HEURISTIQUES INFORMATISEES

Mémoire de fin d' étude
présenté par **Marie GEVERS**
épouse **d' UDEKEM d' ACOZ**
pour l'obtention du grade
de Licenciée et Maître en
Informatique.

Promoteur :**M. NOIRHOMME-FRAITURE**

Copromoteur :**J. BARRETO**

Année académique 1988-1989

Facultés Universitaires Notre-Dame de la Paix
Institut d'informatique
rue de Bruxelles, 61, B-5000 **NAMUR**
Tél: 081-22.90.61 Télex: 59222 facnam-b Téléfax: 081-23.03.91

L'ANALYSE CLADISTIQUE : PROBLEMES ET SOLUTIONS HEURISTIQUES INFORMATISEES

GEVERS MARIE épouse **D'UDEKEM D'ACOZ**

Résumé

Ce mémoire a pour but d'étudier une utilisation non-conventionnelle de l'informatique : celle relative à l'inférence d'arbres phylogénétiques dans l'école cladiste .

Il commence par décrire le problème d'optimisation tel qu'il se pose concrètement en analyse cladistique (revue de la littérature). Sur la base de cette description, il dégage ensuite la structure abstraite du problème , en utilisant une comparaison avec les langages formels.

Puis, il envisage des moyens concrets mis en oeuvre pour manipuler ce modèle formel : il se concentre sur des solutions heuristiques proposées et donne ,notamment, les spécifications de la méthode du "branch swapping".

Abstract

The main goal of this dissertation is the study of a non conventional application of informatics : the inference of phylogenetic trees following the cladistic school.

The first part describes the nature of an optimisation problem, as concretly presented in cladistic analysis (survey of the bibliography). Based on this description, the abstract structure of the problem is highlighted as well as the relation with formal languages.

In the second part, concrete tools to deal with this formal model are presented : particularly, the heuristic solutions proposed are described and the details of the "branch swaping method" are studied.

Mémoire de licence et maîtrise en Informatique
Juin 1989
Promoteur : **M. NOIRHOMME-FRAITURE**
Copromoteur: **J.BARRETO**

Ma très vive reconnaissance s'addresse à Monsieur J. Barreto, Docteur en Sciences Appliquées, qui a dirigé ce mémoire en me faisant profiter de sa formation pluridisciplinaire et m'a suggéré de fructueux thèmes de réflexion et d'approfondissement. Je voudrais témoigner ici du fait que sa compétence intellectuelle n'a d'égale que sa disponibilité et sa gentillesse.

Je tiens aussi à dire ma très cordiale gratitude à Monsieur Y. Coppens, Professeur au Collège de France, qui m'a suggéré le sujet de cette étude .En effet, cette analyse qu'il m'a proposée rencontrait un intéret personnel remontant à l'enfance, m'a enthousiasmée et m'a procuré réellement l'exhaltation d'"apprendre".
Monsieur Coppens a eu la grande amabilité de me prêter un oreille attentive et bienveillante, au cours d'entrevues et de nombreuses conversations téléphoniques.

J'ai une très grande dette envers Monsieur J. Felsenstein, Professeur à l'Université de Washington : il m'a transmis les codes-sources de ses programmes. Sans lui, l'essentiel de la seconde partie de ce mémoire aurait été tout simplement impossible !

Monsieur C. Cherton ainsi que Madame M. Noirhomme-Fraiture et Monsieur J.-C. Hainaut , tous les trois, professeurs aux Faculté Notre-Dame de la Paix à Namur, se sont prêtés à des discussions critiques de ce travail : je leur en suis très reconnaissante.

Je voudrais aussi remercier Monsieur P. Tassy, Professeur de Paléontologie à l'Université Pierre et Marie Curie (Paris VI), qui m'a fourni de précieuses informations et à contribué à ma documentation.

R.L. Pagano, doctorante et boursière, m'a fait profiter de sa collaboration : qu'elle trouve ici l'expression de ma gratitude.

Merci aussi à tous ceux qui m'ont transmis des articles et des renseignements relatifs à l'analyse cladistique : je veux parler, en particulier, de Madame R. Orban, Professeur à l'U.L.B. de Monsieur E. Serusiaux, Chercheur qualifié au F.N.R.S. et de Monsieur M. Laloux, assistant aux F.N.D.P.

Enfin, je voudrais exprimer mon affectueuse reconnaissance à Benoît , mon fils aîné, qui, avec beaucoup de compétence dans l'utilisation du traitement de texte et de persévérance, a contribué à la dactylographie de ce mémoire.

Et pour terminer, je remercie mon cher époux qui m'a soutenue tout au long de cette étude (comme des précédentes...), m'a fait profiter de son esprit critique et m'a offert le voyage à Stokholm pour assister au " Willi Hennig Society Meeting VII ".

TABLE DES MATIERES

CHAPITRE 4 . INTERET DES CLADOGRAMMES.

SECONDE PARTIE	Etude approfondie et comparée d'un algorithme heuristique d'inférences d'arbres phylogénétiques basée sur le principe de parcimonie.

CHAPITRE 5 : INTRODUCTION.

AVANT PROPOS.

La littérature relative à l'analyse cladistique est volumineuse. Ce travail n'a pas la prétention d'être exhaustif. Par souci de concision, j'ai volontairement laissé dans l'ombre certains sujets (pondération des caractères , indice de consistence, arbre-consensus...) et peu détaillé certains autres (théories de l'évolution ...).

Cette littérature est très largement anglo-saxonne. Ceci m'a incitée à conserver dans mon mémoire des termes anglais (tout en en donnant la traduction française). D'autre part, par souci de fidélité, j'ai laissé beaucoup de citations dans leur langue originale (pour autant, toutefois, que le contexte en français soit suffisamment explicite).

Cette littérature forme un tout dont les différents éléments sont étroitement liés. Ceci m'a obligée, dans la première partie de ce mémoire, à faire de nombreuses références en avant et m'a incitée à fournir au lecteur un index de vocabulaire.

Et enfin, cette littérature prend toujours comme référence l'ouvrage de Hennig (1966). Moi de même, dans différents paragraphes relatifs à la revue de la littérature, j'ai commencé par citer cet auteur. Ce faisant, je poursuis le double but de fournir au lecteur une connaissance des idées originelles de l'école cladiste et de lui permettre de juger, le cas échéant, de l'évolution accomplie dans la suite par les disciples de Hennig .

Afin de rendre ce travail accessible aussi bien au biologiste qu'à l'informaticien, j'ai généralement défini le vocabulaire utilisé de manière exhaustive. D'autre part, les termes variant souvent selon les auteurs, je me suis efforcée de donner la correspondance entre ceux-ci. Ce mémoire se réfère presque constamment à la théorie des graphes : j'ai donc consigné dans un appendice (I) les éléments nécessaires au lecteur non familiarisé avec cette théorie. Un autre appendice (II) fournit des notions de vocabulaire emprunté à l'algorithmique.

Pour permettre un lecture non-chronologique de cette étude, j'ai pris l'option d'établir de nombreux renvois connectant entre eux différents paragraphes. Ceci a comme conséquence certaines redites dont je prie le lecteur de m'excuser.

Celui qui possède un bagage en informatique et ne veut pas s'embarrasser de considérations biologiques peut commencer à lire ce travail au paragraphe 2.3 (codage des caractères) et ne pas s'appesantir sur le paragraphe suivant (source). Il peut également passer le chapitre 4 ,à moins qu'il ne soit à la recherche de sujets de réflexion... D'autre part, au niveau de la bibliographie , il pourra repérer les références relatives à l'informatique, aux mathématiques ou à la théorie des langages formels parce qu'elles sont mises en évidence par une astérisque.

INTRODUCTION

L'informatique tend actuellement à envahir tous les domaines.

Beaucoup de problèmes concrets et réels sont maintenant largement reconnus comme étant du ressort de cette discipline. Ainsi en est-il, par exemple de la publication assistée par ordinateur, de la comptabilité, de différentes gestions (stocks, cabinets médicaux ...) etc...

Ce mémoire analyse les possibilités d'informatisation dans un domaine biologique très particulier ,peu familier à la plupart des informaticiens : celui de l'inférence d'arbres phylogénétiques dans l'école cladiste* encore appelée (Farris 1986 b: 15) école de systématique phylogénétique.

Cette dernière se démarque nettement des autres écoles de systématique, à savoir celles des évolutionistes et celle des phénéticiens. Elle a pour fondateur Willy Hennig. La référence de base des cladistes est en effet l'ouvrage que cet auteur écrivit en allemand dès 1950 mais qui ne connut un réel rayonnement que depuis 1966, date de sa traduction en anglais sous le titre "Phylogenetic Systematics".

Dans un arbre phylogénétique selon l'école cladiste, les taxons observés sont des feuilles et les noeuds sont des ancêtres hypothétiques. Un tel arbre est construit sur base des synapomorphies (partage d'états dérivés de caractères) et répond à un critère d'optimisation.

Un cladogramme (arbre phylogénétique enraciné) permet d'étudier le processus de l'évolution (Farris,1986 b & Platnick, 1986) et constitue un "système de référence général" (Tassy ,1988) dont des classifications peuvent être déduites ,sur base de groupes monophylétiques.

L'informatisation de l'analyse cladistique connait actuellement des développements très importants. Mais, dans le même temps, les algorithmes utilisés par les programmes ne sont que très partiellement décrits.

---

* Comme le signale Tassy (1986, p.87), l'usage des termes "cladiste" et "cladistique" est postérieur à Hennig. Ces mots sont "dérivés du mot « clade » qui désigne selon J. Huxley (1957) une unité monophylétique, quoique Cuénot (1840) ait conçu le mot antérieurement avec un sens opposé ...".

La démarche suivie dans ce travail sera typiquement informatique et comprendra deux étapes: l'une concernera essentiellement la définition du problème et la seconde, les solutions.

En effet, la première partie commencera par décrire le problème tel qu'il se pose concrètement en analyse cladistique : elle fera une revue de la littérature concernant tout d'abord les groupes biologiques à analyser (chapitre 1), ensuite les caractères à attribuer à ceux-ci (chapitre 2) et enfin les différentes instanciations du problème (début du chapitre 3).
Sur la base de cette description, la structure abstraite ou modèle formel du problème pourra être dégagée (fin du chapitre 3).Cette première partie s'achèvera par la mise en évidence de l'intérêt des cladogrammes (chapitre 4).

La seconde partie envisagera des moyens concrets mis en oeuvre pour manipuler la strucure abstraite définie antérieurement.
Elle se concentrera sur des solutions heuristiques apportées au problème particulier de la construction d'arbres de Wagner (au sens large). Après avoir justifié son étude et brièvement présenté le programme MIX de Felsenstein (chapitre 5) , elle analysera de manière de plus en plus détaillée l'algorithme de recherche des topologies les plus parcimonieuses utilisée par ce programme (chapitre 6). Ensuite elle le situera parmi les autres programmes d'analyse cladistique et les comparera, autant que faire se peut, au logiciel PAUP de Swofford (chapitre 7).

Et ce mémoire s'achèvera par quelques conclusions relatives à l'informatisation de l'analyse cladistique et par des propositions d'études futures dans ce domaine.

PREMIERE PARTIE	L'analyse cladistique : revue de la littérature et définition du problème.

CHAPITRE 1. LES TAXONS A ANALYSER.

Ainsi donc, la première partie de ce mémoire s'intéresse à la définition du problème posé en analyse cladistique.
Elle commence par une revue de la littérature relative à ce sujet. Et en tout premier lieu, elle prend en compte les données du problème . Elle s'interroge d'abord sur les groupes biologiques pouvant être l'objet de l'analyse.

1.1 DEFINITIONS PRELIMINAIRES.

Taxon : "Simpson en donne la définition suivante : "un taxon est un groupe d'organismes reconnus en tant qu'unité formelle à chacun des niveaux d'une classification hiérarchique" (Janvier et al., 1980).

Operational taxonomic units ou OTUs : terme introduit par Sokal et Sneath (1963) et utilisé dans la méthode de parcimonie (cf.§ 3.2.1.1.1) pour désigner les taxons soumis à l'analyse. Ces taxons seront les feuilles de l'arbre phylogénétique construit comme résultat ("terminal taxa").

Hypothetical taxonomic units ou HTUs : terme introduit par Farris (1970, p.85) et utilisé dans la méthode de parcimonie pour désigner, dans un arbre phylogénétique, les ancêtres hypothétiques communs à des groupes d'OTUs.

Evolutionary units ou EUs : terme utilisé en analyse de compatibilité pour désigner "les variétés d'organismes présents dans la collection à l'étude" (Meacham, 1984, p.26). Il correspond donc à OTUs en analyse de parcimonie.

1.2 QUELS TAXONS INTRODUIRE DANS UNE ANALYSE CLADISTIQUE ?

Dans une analyse cladistique peuvent figurer aussi bien des taxons fossiles que des taxons actuels. Dans la mesure des possibilités, il est chaleureusement recommandé d'utiliser simultanément ces 2 variétés de taxons. Gauthier, Kluge et Rowe (1988, p.105), preuve à l'appui, donnent le conseil suivant : "We urge systematists to evaluate fairly all of the available evidence." Et Kluge (1989 :10) explique : "The principle of total evidence is an important maxim in phylogenetic systematics..."

En ce qui concerne les fossiles, il faut toutefois garder à l'esprit qu'il existe une réelle difficulté à préciser les critères d'identification de l'espèce. Car comme le soulignent Turner et Chamberlain (1989 : 127) : "There is no consistent correlation between speciation and morphological change."

D'autre part, comme le soulignent de Queiroz et Donoghue (1988, p.325 à 327), les méthodes de systématique phylogénétique sont basées sur la prémisse qu'il existe une structure hiérarchique de relations ("... nested hierarchical pattern of relationships") entre les entités à étudier. Il est donc exclu d'appliquer ces méthodes pour analyser des organismes se croisant réellement (car les relations entre ces derniers forment un réseau) (cf. Hennig, 1966, p.18 et 19). Par contre ,des taxons spécifiques ou supraspécifiques peuvent toujours être soumis à l'analyse cladistique. Une question pertinente est la suivante : quelles sont les entités les plus fines groupant des individus sexués* et étant susceptibles de former les feuilles d'un arbre phylogénétique ? La réponse à cette question est fonction de la définition adoptée pour le concept d'espèce.

1.3 ENTITE ULTIME POUVANT ETRE INTRODUITE DANS UNE ANALYSE CLADISTIQUE ET PROBLEME DE L'ESPECE EN PHYLOGENETIQUE SYSTEMATIQUE.

1.3.1 **Sources**.

Selon Hennig (1966 , p. 29) , les espèces sont des groupes d'individus qui sont interconnectés par des relations impliquant des phénomènes de "reproduction". Toujours selon lui, les relations phylogénétiques (c-à-d celles mises en évidence dans un arbre phylogénétique orienté) sont celles existant entre espèces.

---

* Le cas des organismes asexués ne sera pas évoqué ici mais il est pris en compte par de Queiroz et Donoghe (1988).

Un groupe monophylétique est un groupe d'espèces descendant d'un ancêtre commun unique et qui comprend toutes les espèces issues de cet ancêtre commun (traduit de Hennig, 1966, p.73). Une remarque de Hennig (1966, p.71) indique qu'il considère que l'ancêtre commun fait partie intégrante du groupe monophylétique. Et Hennig d'ajouter (1966, p.146) qu'un tel groupe a donc une histoire propre et correspond à une réalité.

1.3.2 **Une synthèse du problème de l'espèce en systématique phylogénétique.**

La définition de l'espèce citée ci-dessus ne fait pas l'unanimité : le concept d'espèce a déjà fait couler beaucoup d'encre et encore actuellement la controverse à ce sujet est particulièrement vive. Comme ce mémoire ne peut être trop volumineux, je ne citerai pas ici de référence bibliographique et je me contenterai de résumer les conclusions d'une intéressante synthèse réalisée par de Queiroz et Donoghue (1988) à propos du "problème de l'espèce" en systématique phylogénétique.
Ces auteurs considèrent que, dans une analyse cladistique, trois différents concepts d'espèces basés sur des processus naturels distincts, peuvent être utilisés valablement mais avec des implications différentes.

a. Concepts de l'espèce basés sur l'interfécondité ("interbreeding").

Ce concept biologique, qui ne peut malheureusement s'appliquer aux organismes asexués, peut avoir 2 acceptions.

Au sens strict, le terme espèce sera réservé à chaque population* constituant **effectivement** une communauté de reproduction (qu'elle soit ou non monophylétique).
Cette première utilisation du mot espèce se heurte à des difficultés pratiques. En effet, d'une part il est souvent extrêmement difficile de fixer les limites d'une communauté de reproduction effective car, comme l'explique de Queiroz et Donoghue (1988, p.329) : "...the degree of gene flow varies in

---

* de Queiroz et Donoghue (1988, p.326) définissent les populations comme suit : "...units within which interbreeding between organisms of different subunits is sufficient such that the relationships among these subunits are reticulate while relationships among the units themselves are predominantly diverging".

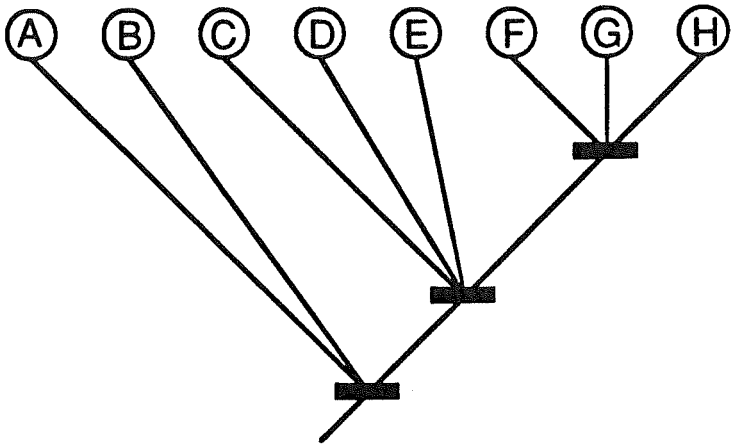


fig. 1.1 Concept d'espèce basé sur l'interfécondité au sens strict.
Un cladogramme de 8 populations (A-H)
(extraite de de Queiroz et Donoghue, 1988, p.328).

space and time and there need be no correspondence between interbreeding and morphological or ecological divergence...".
D'autre part, le nombre d'espèces ainsi obtenu serait beaucoup plus important que celui des espèces couramment reconnues.

Selon la version élargie ou standard, le nom d'"espèce" est donné à un groupe d'organismes constituant **potentiellement** une communauté de reproduction. Une telle espèce peut se composer de plusieurs communautés de reproduction effectives ou populations (par exemple A-E et F-H dans la figure 1.2).
de Queiroz et Donoghue (1988, p.330) soulignent que ce concept d'espèce présente les intérêts suivants : "...il tente d'incorporer l'idée que les espèces existent pendant une certaine durée au cours de l'évolution ... De plus, la perte de la potentialité de se croiser garantit que les entités fonctionnent comme des unités évolutives distinctes". Mais il y a aussi des inconvénients à employer le mot espèce avec cette sémantique élargie. Tout d'abord, une telle espèce peut n'être qu'une collection artificielle sans fondement biologique : elle peut ne pas être monophylétique (cf. groupe A-E dans la figure 1.2) et ne pas correspondre réellement à une communauté de reproduction.
Ensuite, il est difficile de déterminer, sur bases de comparaisons de la morphologie, du comportement ou de l'écologie, quels organismes "pourraient" se croiser.

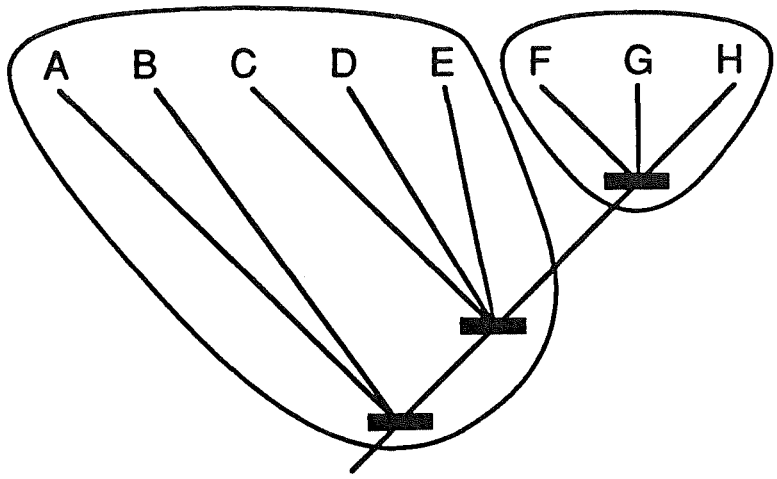


fig. 1.2 Concept d'espèce basé sur l'interfécondité au sens large.
Un cladogramme de 8 populations séparées.
Les organismes des groupes de populations encerclés (A-E et F-H) ont la potentialité de se croiser. Des croisement sont impossibles entre individus de cercles différents.
(extraite de de Queiroz et Donoghue , 1988, p.329).

b. Concepts de l'espèce basé sur la monophylie, reposant elle-même sur le processus naturel de "communauté d'origine" ("common descent").

Ce concept s'est développé récemment en réponse à la remarque suivante (cf. Rosen, 1978-1979 et Bremer et Wanntorp, 1979): il se pourrait que la compatibilité de reproduction se distribue comme une mosaïque parmi les populations descendant d'un ancêtre commun de telle façon que, comme un caractère primitif conservé, elle soit sans information à propos du caractère récent de l'ancêtre commun. Et dès lors si un ensemble reçoit le statut d'espèce sur base de la capacité de croisement fécond, il se pourrait que certaines espèces ne soient pas monophylétiques (de Queiroz et Donoghue, 1988, p.318). Selon ce nouveau concept, le terme espèce est réservé à un certain ensemble de groupes monophylétique, qu'il corresponde ou non à une communauté de reproduction : il désigne donc un rang donné dans la hiérarchie de relations phylogénétiques. On peut par exemple choisir d'appeler "espèce" le groupe monophylétique le plus inclus (F-H dans la figure 1.2).

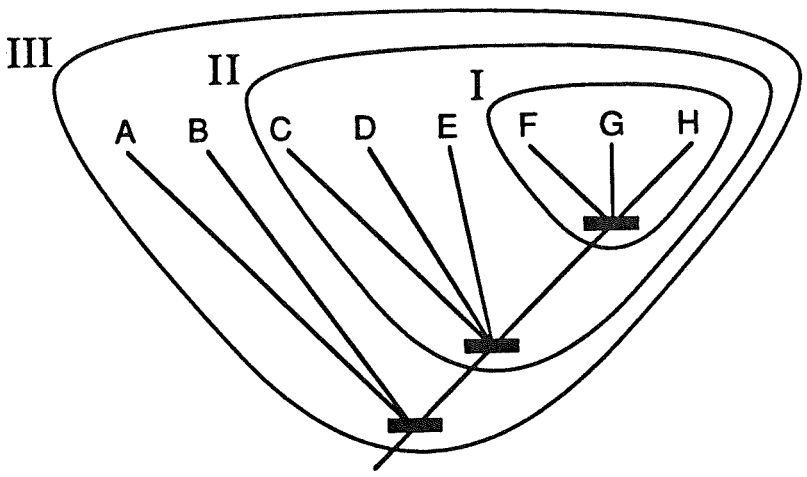


fig. 1.3 Concept d'espèce basé sur la monophylie.
Un cladogramme de 8 populations montrant 3 groupes monophylétiques (I-III).
(extraite de de Queiroz et Donoghue 1988, p.331.)

Mais quel que soit le groupe monophylétique choisi comme espèce, il ne sera pas possible d'assigner, à toutes les entités terminales, un taxon monophylétique dont le rang est l'espèce (ce qui viole une ancienne convention). D'autre part, lorsqu'on considère l'utilité des rangs, une objection plus générale à la définition de l'espèce comme un niveau dans la hiérarchie de groupes monophylétique surgit . Comme le soulignent de Queiroz et Donoghue (1988, p.333) en effet : "...categorical ranks add no information about monophyly that is not already contained in a cladogram...". Et les auteurs de poursuivre le raisonnement : si les rangs ne sont pas utiles, ils pourraient être abandonnés et par conséquent le terme espèce ne serait plus employé pour désigner un ensemble monophylétique donné et il pourrait alors être réservé pour définir une communauté basée sur l'interfécondité.

c. Concept disjonctif de l'espèce.

D'après ce concept, les espèces pourraient être soit des populations (basées sur l'interfécondité), soit des groupes monophylétiques de populations. Ce concept allie donc 2 processus naturels distincts. Selon l'analyse de de Queiroz et Donoghue (1988), cette acception peut créer davantage de problèmes qu'elle n'en résoud car elle résulte en taxons spécifiques qui ne sont pas comparables.

1.3.3 **Conclusion.**

Cette synthèse de de Queiroz et Donoghue (1988) met en évidence que selon la définition choisie pour le concept d'espèce, les entités les plus fines susceptibles d'être les feuilles d'un arbre phylogénétique pourront être des espèces et/ou des populations d'organismes sexués.

CHAPITRE 2.LES CARACTERES.

Pour continuer cette revue de la littérature cladistique relative, tout d'abord, aux données du problème de l'inférence d'arbres phylogénétiques, on va s'intéresser maintenant aux caractères utilisés pour décrire les taxons à introduire dans l'analyse.

2.1.DEFINITIONS RELATIVES AUX CARACTERES.

Il existe, dans la littérature cladistique, un très grand nombre de définitions différentes relatives aux caractères. Ceci est notamment dû aux progrès de l'analyse phylogénétique elle-même : cette dernière allant en s'enrichissant, les définitions doivent elles-mêmes évoluer vers davantage de généralités, d'une part et davantage de précisions, d'autre part. Une synthèse peut être trouvé au tableau 2.1.

Le cadre général des définitions retenu ici est celui de Goldman (1988, p.70 et 71). Cet auteur fait très pertinemment la distinction explicite entre observation et représentation dans la matrice de données. Il appelle caractère, un trait ("feature") observable* du taxon, choisi pour être utilisé dans l'analyse. Les états du caractère sont les valeurs que peuvent prendre ce trait. Ils sont évidemment définis par l'observateur. Ainsi par exemple, le caractère nommé couleur du pelage pourra avoir les états suivants : brun, gris, noir, roux. Chaque caractère est codé de façon à être représenté par une ou plusieurs variables numériques**. Les valeurs d'une variable seront appelées états de la variable. Pour les problèmes d'inférence d'arbres phylogénétiques pris en compte dans ce mémoire (cf.tableau 3.1) les variables sont obligatoirement discrètes***. Dans les cas de base (cf. par exemple la description du programme MIX aux chapitres 5 & 6), le domaine d'une variable est l'ensemble des entiers positifs. Le passage d'un état de la variable (par exemple 0) à un autre (par exemple 1) peut alors s'interpréter comme un changement évolutif ou pas (step). (Ce dernier concept a été introduit par Camin et Sokal (1965) et par Farris et al.

---

* Typiquement en analyse cladistique, un trait observable pourra être morphologique, ou comportemental, ou génétique, ou encore moléculaire (cf.paragraphe 2.1.1).

** Il existe une exception à cette règle, mentionnée au paragraphe 3.2.1.1.1.2 : dans le problème de Steiner, les variables seront nominales.

*** Certaines méthodes cependant sont relatives à des variables continues, notamment des fréquences de gènes.(cf.par exemple Swofford et Berlocher, 1987).

(1970) à propos du changement d'état pour un caractère discret). Il faut encore noter que les variables de l'analyse cladistique peuvent être l'objet de pondération mais cet aspect ne sera pas développé dans la présente étude.

La définition que donne Goldman d'un caractère est tout à fait générale puisqu'elle s'applique à toutes les données, qu'elles soient discrètes ou continues. Ces dernières posent cependant un gros problème en analyse phylogénétique. Par rapport aux données continues, on rencontre deux tendances actuellement parmi les cladistes. Certains en effet considèrent que les données quantitatives sont impropres à être traitées par analyse cladistique. C'est ainsi que Pimentel et Riggins (1987, p.201) écrivent : "Continuously varying quantitative data are not suitable for cladistic analysis because there is no justifiable basis for recognizing discretes states among them." Et ces auteurs poursuivent (1987, p.208) "A measurement such as length of a bone pertains only to a linear dimension of the bone not the total bone. In this sense, dimensions are indirect measures of a feature and sufficiently vague to question what is appropriate for tests of homology." Selon le compte-rendu de Coddington (1987, p.181), Mitter partage le scepticisme de Pimentel et Riggins (1987) à propos de la conversion de données quantitatives en variables discrètes. Mais certains tentent de relever ce défi. Ainsi Archie (1985) a développé une technique appelée "generalized gap-coding" pour transformer les caractères continus en variables discrètes. Cette méthode a été récemment améliorée et complétée par Goldman (1988). La technique d'Archie (1985) modifiée par Goldman vient d'être utilisée par Cranston et Humphries (1988). D'autre part, Chamberlain fit, au 7ème Congrès de la Société Willi Hennig (1988), une communication intitulée "Methods for coding continuous morphological characters for cladistic analysis".
Cette digression étant faite à propos de données continues, le terme "caractère" se rapportera dans la suite de ce mémoire à un caractère discret.

Les définitions données ci-dessus peuvent être raffinées en y adaptant les catégories distinguées par Day et Sankoff (1986) et Day ,Johnson et Sankoff (1986). On qualifiera alors une variable de l'analyse cladistique ou un caractère (discret) soit de <u>binaire</u> (si elle/il ne peut prendre que deux états), soit de <u>sans contrainte</u> ("multistate"). Day, Johnson et Sankoff font aussi la distinction entre qualitatif et cladistique. Le terme <u>qualitatif</u> sera réservé un caractère (discret) dont les états forment un ensemble non ordonné, auquel aucune structure n'est imposée. A la suite des trois auteurs cités ci-dessus, on pourra appeler <u>cladistique</u> un caractère discret à la fois ordonné et dirigé [c'est-à-dire enraciné]. Tout comme Meacham (1984), on considèrera aussi la catégorie des caractères discrets <u>ordonnés mais non dirigés</u>. (Cette dernière variété n'est pas prise en compte par Day et al.)

Un caractère cladistique de Day, Johnson et Sankoff (1986) correspond en réalité au "caractère" tel que défini formellement jadis par Farris, Kluge et Eckardt (1970, p.172) en référence à Hennig (1966) : "A character ('transformation series' of Hennig)

is a collection of mutually exclusive states (...'characters', 'character states', or 'stages of expression' of Hennig) which
a) have a fixed order of evolution such that
b) each state is derived directly from just one other state and
c) there is a unique state from which every other state is eventually derived."

Il est évident que l'ordre et la direction (ou polarité) d'évolution des états d'un caractère cladistique peut être représenté par un arbre enraciné (cf. "character state tree" de Camin et Sokal, 1965) dans lequel chaque sommet représente un état observé et chaque arête une transformation orientée possible.

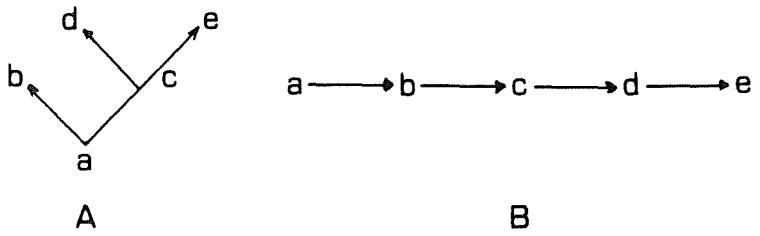


Fig.2.1 arbre enraciné d'états du caractère C.
Les états de ce caractère sont a,b,c,d,e.
A : série de transformations branchée.
B : série de transformations linéaire.

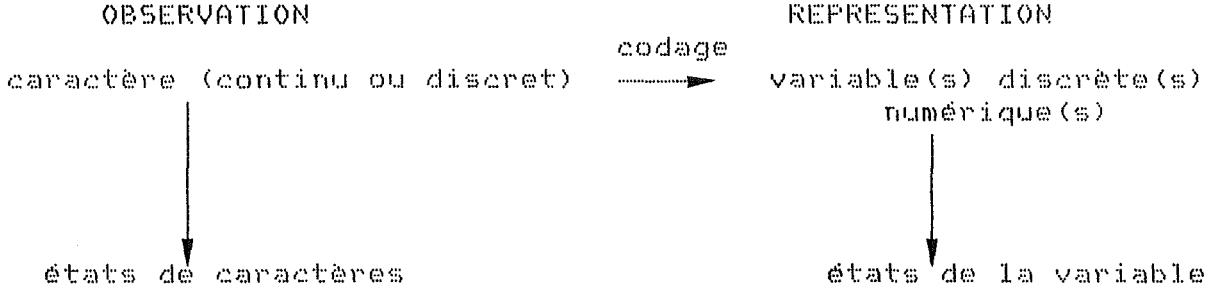
A partir de cette représentation, la classification des états en ancestraux (ou "plesiomorphe" de Hennig [1966, p. 89] ou encore primitifs) et dérivés (ou "apomorphes" de Hennig) s'explique aisément. Ainsi par exemple dans la figure 2.1.A, l'état c est ancestral **par rapport** à l'état e mais il est dérivé **par rapport** à l'état a . Il est clair que ces dernières définitions sont **relatives** à un point de référence. Toutefois, au niveau de la racine cladistique, tous les états de tous les caractères sont ,par définition, primitifs.

A noter encore que pour une série de transformations, la distinction entre "linéaire" et "branchée" est souvent faite dans la littérature cladistique. Cette distinction est relativement arbitraire puisqu'elle correspond à deux cas particuliers d'arbres.

D'autre part, c'est un arbre non enraciné qui correspondra à un caractère ordonné, mais non dirigé.

TABLEAU 2.1. Synthèse de définitions relatives aux caractères, en référence à différents auteurs.

1. Cadre général (Goldman 1988).



2. Catégories applicables à un caractère discret (de l'analyse cladistique) :

* - binaire (si 2 états) [Day et al, 1986]
  - sans contrainte (si plus de 2 états) [" "]

* - qualitatif (c-à-d non ordonné) [" "]
  - cladistique (c-à-d ordonné et dirigé [" "]
  - ordonné et non dirigé [Meacham, 1984]

N.B.: qualitatif=non-additif pour Platnick(1988).

3. Quelques identités et correspondances importantes

* caractère cladistique (Day et al, 1986) = série de transformations [(Hennig, 1966) & (Goldman ,1989)] = caractère (Farris et al,1970)

arbre (enraciné) d'états de caractère (Camin et Sokal, 1965) et (Farris et al, 1970).

* caractère ordonné et non dirigé (Meacham, 1984)

arbre non enraciné d'états de caractère (Farris et al,1970) arbre (Goldman, 1989).

2.2. ANALYSE DES CARACTERES DISCRETS.

2.2.0. **Introduction.**

L'analyse des caractères discrets constitue un point crucial puisqu'il s'agit d'un préalable obligatoire à l'analyse cladistique proprement dite et qu'elle conditionne la valeur de cette dernière. Tassy et Darlu (1988 :295) affirment d'ailleurs : "N'oublions pas ... qu'en morphologie le stade le plus important de l'analyse réside dans la construction de la matrice de caractères."

Comme le souligne Meacham (1984, p.27), cette analyse, pour chaque caractère, peut présenter un nombre variable d'étapes selon la méthode d'analyse utilisée ultérieurement .

i. Il faut tout d'abord obligatoirement inventer le caractère et en repérer les états.
ii. Il faut ensuite, pour beaucoup de méthodes, déterminer l'ordre à établir entre les états de ce caractère.
iii. Enfin, on peut encore faire l'hypothèse supplémentaire de la direction (polarisation) de l'évolution des états pour ce caractère.

2.2.1. **Polarisation des états.**

Cette dernière étape de l'analyse des caractères peut reposer sur différents critères, de valeurs diverses, et dont certains sont actuellement très controversés. Dans Stevens (1980) et Arnold (1981) peut être trouvé un aperçu de la littérature déjà ancienne au sujet de ces critères. Parmi ceux-ci, les plus fréquemment cités à l'heure actuelle sont les suivants :

1. Critère de l'outgroup
2. Critère ontogénique
3. Critère de l'ingroup
4. Critère paléontologique
5. Critère fonctionnel.

Les deux premiers éléments de cette liste sont incontestablement les plus utilisés actuellement. Les deux suivants, s'ils sont employés dans des études relativement anciennes, ont tendance à tomber en désuétude complète. Quant au dernier, il est cité pro forma.

2.2.1.1. Critère de l'outgroup.

Définition préliminaire : outgroup ou extragroupe : groupe apparenté à l'ensemble des OTUs devant être analysé c-à-d. l'ingroup et pouvant lui-même comporter un ou plusieurs OTUs.

Remarque préliminaire : L'outgroup doit être clairement distingué de la racine cladistique ! Mais il se trouve que dans la littérature cladistique, les états de l'outgroup (lorsqu'il ne comporte qu'un seul OTU) sont couramment assimilés à ceux de la racine cladistique. Ceci peut prêter à confusion. Il me paraît utile de souligner qu'en toute généralité, l'OTU désigné comme outgroup et la racine cladistique peuvent évidemment avoir des états différents. L'exemple de Felsenstein (1983, p.315) illustre d'ailleurs bien ce fait.

Le critère de l'outgroup est considéré comme le meilleur par la plupart des cladistes. Malheureusement, il en existe plusieurs versions.

Comme le fait remarquer Farris (1982a,p.333), celui qui veut bien lire entre les lignes en trouve une formulation dans Hennig (1966, p.99) ,reprenant Maslin (1952) : "If one extreme of a morphocline resembles un condition found in the less modified members of related groups of the same rank, this extreme is primitive". Hennig lui-même ne mentionne pas le mot "outgroup", fait remarquer que Naef appelait le principe cité ci-dessus "systematic character precedence" et ajoute encore que cette formulation est peut-être équivoque ...

En 1981, Watrous et Wheeler disent avoir constaté dans la littérature beaucoup d'interprétations erronnées du critère de l'outgroup, et en proposent la "règle opérationnelle" suivante : "Si un état d'un caractère se rencontre à la fois dans l'ingroup et dans l'outgroup et que d'autres états n'existent que dans l'ingroup, alors l'état commun à l'ingroup et à l'outgroup doit être considéré comme primitif". Cet énoncé est très voisin de celui de Hennig, comme le note Farris (1982a).

C'est d'ailleurs, cette règle qu'utilisent, de manière très explicite, mais sans citer de source, Andrews et Martin (1987,

p.107 et 108) dans leur exemple didactique. Dans ce dernier, se présente aussi le cas non prévu par la règle où l'état dans l'outgroup est différent de l'état dans l'ingroup. Mais curieusement, les auteurs restent muets sur la facon dont il faut régler ce cas de figure que cependant ils évoquent.
Le critère de l'outgroup, tel que défini dans Stevens (1980, p.337) est une variante de la règle opérationnelle de Watrous et Wheeler (1981), qui nécessite 2 outgroups.

L'article de Watrous et Wheeler est critiqué avec beaucoup d'acrimonie par Farris (1982a). Ce dernier fait remarquer que leur "règle opérationnelle", même si elle peut s'avérer correcte pour des cas simples, est erronée en toute généralité : ("...their procedure is itself erroneous"). Outre le cas de figure mentionné ci-dessus, elle est inapplicable par exemple si au sein de l'outgroup le caractère varie (cf. Maddison et al. 1984). En réalité, explique Farris (1982a), la règle de l'outgroup ne peut se comprendre que dans le cadre de l'analyse de parcimonie (cf.paragraphe 3.2.1.1.1) dont elle constitue une application.
De plus, comme l'ont montré Clark et Curran (1986) à propos de la parcimonie de type Wagner (sensu stricto), une analyse simultanée de l'outgroup et de l'ingroup est supérieure à celle réalisée en 2 étapes (cf. la procédure de Maddison et al. 1984 obtenant un arbre le plus parcimonieux globalement mais présumant la monophylie de l'ingroup). En effet, une analyse simultanée fait moins d'hypothèse a priori que toute autre technique : elle suppose uniquement que la racine cladistique est basale par rapport à l'ingroup. Et dès lors si l'ingroup n'est pas monophylétique, l'analyse conjointe de l'ingroup et de l'outgroup donnera une solution plus parcimonieuse que la méthode en 2 étapes.
En outre, Farris (1970, p.89-90) et Meacham (1984) recommandent de commencer l'analyse par un arbre non-enraciné puis de l'enraciner ensuite.
Précisément, dans les programmes analysés dans la présente étude (MIX 3.1 et PAUP 2.4) le critère de l'outgroup* est pris en compte** de la facon suivante : dans un premier temps, l'arbre non-enraciné le plus parcimonieux comportant à la fois l'ingroup

---

* L'enracinement par outgroup ne doit pas être confondu avec l'enracinement de Lundberg (1972). Comme l'explique Swofford (1985, p. 3-8) qui offre cette option dans PAUP : dans cette méthode, un arbre non enraciné le plus parcimonieux, ne comportant que les OTUs devant être analysés, est d'abord construit. Ensuite cet arbre est enraciné à l'endroit où un ancêtre présumé (c'est-à-dire un OTU dont les états de tous les caractères sont primitifs) ou un outgroup (...) s'ajouterait à l'arbre.

** Le critère n'est pris en compte que si l'arbre peut être construit non-enraciné, c'est-à-dire si tous les caractères sont de type Wagner (sensu stricto) (c.f.paragraphe 3.1.1.1.2.) et d'état primitif inconnu.

et l'outgroup est construit à partir de la matrice de variables, et, dans un second temps, cet arbre est enraciné par outgroup: il est donc réorienté de telle façon que l'outgroup devienne le groupe-frère des autres OTUs, étant entendu que la racine cladistique, surajoutée à l'arbre non enraciné, vaut à la fois pour l'ingroup et l'outgroup. Si l'outgroup est constitué de plusieurs OTUs (cf. PAUP), il se pourrait que cette réorientation soit impossible : il pourrait en effet ne pas être possible de trouver une racine telle que l'ingroup désigné par l'utilisateur soit monophylétique. Cette méthode offre donc de surcroît un test de la monophylie de l'ingroup ! Ceci est illustré par la fig. V.1 qui est un fragment d'"output" de PAUP obtenu à partir de données contenues dans le tableau 7.1, en ayant désigné les taxons alpha et beta comme outgroup.

Ainsi donc, ce critère qui jadis n'était pas formulé de manière assez générale et était utilisé pour polariser les caractères avant la construction "à la main" d'un arbre d'emblée enraciné cladistiquement, est devenu actuellement, grâce à des programmes d'ordinateur, tout à fait général et n'est pris en compte qu'après que l'arbre non-enraciné le plus parcimonieux soit déjà construit. Ceci présente l'énorme avantage d'éviter à l'utilisateur de décider lui-même de la polarité des caractères. "By including a combination of outgroup and ingroup taxa in the analysis, outgroup structure is left to 'float' so that the overal tree is most parsimonious; the results are not tainted by arbitrary or erroneous polarity decisions." (Swofford 1985,p.38) Cet avantage est particulièrement manifeste lorsque l'outgroup comporte plusieurs OTUs, ce qui est recommandé par Maddison et al. (1984, p.99) : "... the ancestral state assessment is more robust the closer and more comprehensive the outgroups ..."
Et Farris (1985, p.293) de conclure : "The value of the outgroup criterion, like the parsimony criterion on which it is based, is simply that it relates preference among hypothesis of plesiomorphy to their relative ability to explain observed similarities among taxa ..."(cf. paragraphe 4.3.2.).

Toutefois , comme le souligne Kraus (1988 : 106), l'utilisation de ce critère peut être problématique , notamment par suite de la difficulté possible à désigner l'outgroup.

2.2.1.2. Critère ontogénique.

Définition préliminaire (extraite du Petit Robert) :

Ontogenèse : développement de l'individu, depuis la fécondation de l'oeuf jusqu'à l'état adulte.

Pour définir le critère ontogénique, la formulation suivante, due à Nelson (1978, p.327) et reprise par Nelson et

Platnick (1981, p.332) semble actuellement unanimement reconnue : "Given an ontogenetic character transformation, from a character observed to be more general to a character observed to be less general, the more general character is primitive and the less general, advanced." Par contre, l'origine et les parentés, dans l'histoire de la biologie, de l'idée véhiculée par cette formulation sont largement controversées. Pour s'en persuader, il suffit de comparer par exemple Løvtrup (1986),de Queiroz (1985) et Wiley (1981). A propos de la direction d'une série de transformation, Hennig (1966, p.95-96) cite déjà un critère analogue : "Criterion of ontogenetic character precedence" et y associe notamment les noms de Naef, Haeckel ainsi que les concepts de loi de récapitulation et de loi biogénétique.
Selon Kraus (1989 : 7), le terme " general" utilisé par Nelson (1978) est ambigu : "... l'état de caractère plus 'general' peut être celui qui émerge le premier au cours de l'ontogenèse ou celui qui, parmi les états ontogéniques, est plus largement distribué au sein des membres du groupe étudié, indépendamment du séquencement ontogénique. Rosen (1982) et Patterson (1982) choisirent la première interprétation, Nelson (1978) et de Queiroz, la seconde."

D'autre part, c'est notamment à propos de ce critère ontogénique qu'une dichotomie s'est produite au sein des cladistes (cf. Kluge, 1985) entre ce que Beatty (1982) a appelé les "pattern cladists" et les "phylogenetic cladists". Les premiers (Nelson, Platnick, Patterson et Rosen) affirment selon Kluge (1985, p.13) que le critère ontogénique est à la fois nécessaire et suffisant à la découverte d'un ordre dans la nature. Les seconds utilisent pour la polarisation des caractères à la fois le critère ontogénique et la méthode de l'outgroup, avec toutefois une prédilection de plus en plus marquée pour cette dernière. Ainsi par exemple, Brooks et Wiley (1985, p.10) considérent l'approche des "pattern cladists" comme "conservatrice". Ils voient le critère ontogénique en rapport à la méthode de l'outgroup comme une partie en regard d'un tout. "Direct observation of ontogeny does not resolve any cases that outgroup comparison fails to resolve, and outgroup comparison does resolve some cases where direct observation fails". Quant à Kluge (1985, p.13) il considère l'approche des "pattern cladists" comme "innacceptable" et il résume ainsi un long plaidoyer contre la prédominance accordée au critère ontogénique : "Dedifferenciation, paedomorphosis (= the phylogenetic loss of adult characters), and the insertion and deletion of developmental stages make it impossible to deduce the genealogical hierarchy from only ontogenetic transformation series". De plus, selon lui, le critère ontogénique n'est pas général. Il affirme en effet (1985, p.24) : "certain major groups of living things (prokaryotes, single-cell eukaryotes) may have no ontogeny".

Cette dernière affirmation de Kluge (1985) a été battue en brèche par Blackmore (1986) et Kociolek et Williams (1987). Prenant comme exemple les diatomées, ces derniers ont démontré

que des transformations ontogéniques peuvent être identifiées au cours de la morphogenèse cellulaire :"Thus we can identify ontogenetic transformations at two levels; at the level of subcellular development at a particular stage in the life history (mitosis, morphogenesis, gametogenesis, etc.) and at a higher level between different stages in the life cycle" (Kociolek et William 1987, p.276).- Il conviendrait donc que la définition du dictionnaire citée ci-dessus soit étendue.- Et ces auteurs (p.281) concluent : "...in unicellular... life cycles, ontogenetic transformations can be... used in determining polarity of character states."

De plus , Kraus (1988) a montré, à partir d'une étude empirique, que l'inconvénient de la pédomorphose n'était pas aussi sérieux que le prévoyaient les considérations théoriques de Kluge (1985). En effet, sur base d'un exemple ,il a mis en évidence la constatation suivante : " ... no effect on tree topology is had by paedomorphic characters until they comprise approximately 50 % or more of the data set..."(Kraus, 1988 : 124). Il conclut donc : "... the requirement for successful application of the ontogeny criterion may be widely met in nature..."

Quant à Nelson (1985), il répond point par point aux critiques de Brooks et Wiley (1985) puis de Kluge (1985) et il repousse, d'un revers de main, les arguments de ses détracteurs. "Analysis of particular examples that have been offered as proof of the themes shows to be flawed and without significance." Et enfin, il souligne la composante philosophique et sociologique des 2 articles auxquels il répond : "I argue that the real substance of the papers is not analysis but philosophy and sociology - that the substance is the cause, not the result, of the analysis." (Nelson 1985, p.39).

La controverse à propos du critère ontogénique demeure donc ouverte.

2.2.1.3. Critère de l'ingroup .

Définition préliminaire :

ingroup ou intragroupe : ensemble des OTUs devant être analysés.

Selon le critère de l'ingroup, l'état du caractère le plus fréquent au sein du groupe étudié ("the common state") est considéré comme primitif.

Ce critère, inexistant dans le traité de Hennig (1966), semble avoir eu une vogue plus grande parmi les botanistes que parmi les zoologistes (cf. synthèse de Stevens,1980, p.335). Une revue de la littérature à ce sujet, textes à l'appui, peut être trouvée dans Watrous et Wheeler (1981). A noter que le critère de l'ingroup a été utilisé par Kluge et Farris (1969, p.5) eux-mêmes et préconisé par des paléontologues comme Delson et al. (1977, p.265) et Eldredge et Tattersall (1975, p.229).

Mais les années passant, il fut peu à peu mis aux oubliettes. Stevens (1980) condamne l'usage de ce critère. Il souligne en effet que la fréquence d'un état d'un caractère dans un groupe dépend de l'histoire évolutive de la lignée et est indépendante de la polarité du caractère. De la même facon, Watrous et Wheeler (1981), après une analyse détaillée et la construction de contre-exemples concluent que doit être abandonné ce principe selon lequel, commun dans l'ingroup signifie primitif. Plus récemment, Meacham (1984, p.35) émet cette même opinion.

2.2.1.4. Critère paléontologique.

Le critère paléontologique (ou géologique) est cité dans Hennig (1966, p.95) : "If in a monophyletic group, a particular character condition occurs only in older fossils and another only in younger fossils, then obviously the former is the plesiomorphous and the latter the apomorphous condition ..." Ce critère jadis préconisé par Kluge et Farris (1969) fut encore utilisé récemment par Skelton et al. (1986) dans leur analyse des hominidés fossiles. Mais ceci paraît être une exception chez les cladistes, car, comme le fait remarquer Wiley (1981, p.148):" The general application of this rule leads to the circular argument that what is older must be primitive and what is primitive must

be older. However, just because a taxon is older and occupies a more primitive phylogenetic position we cannot then assume, a priori, that its characteristics are all plesiomorphic".
Cette attitude générale des cladistes par rapport au critère paléontologique est en parfaite opposition avec celle des évolutionnistes (cf. paragraphe 4.2). Pour ces derniers en effet, "la position chronologique d'un fossile humain sert en grande partie à définir la nature plus ou moins primitive d'un caractère" (Janvier et al.,1980).

2.2.1.5. Critère fonctionnel.

Pour les cladistes purs et durs, des considérations fonctionnelles ne doivent pas intervenir pour déterminer la polarité des caractères. Hennig (1966) n'envisage pas ce critère. D'autre part, Stevens (1980, p.352) après avoir fait une revue de la littérature à ce sujet, conclut : "General ideas about the nature of the evolutionary process, including functional considerations are of course inadmissible ... not least because of the dangers of circulare argument inherent in their use."
Quant à Cracraft (1981, p.28-29), après avoir exposé les arguments de quelques auteurs en faveur de l'utilisation du critère fonctionnel, il déclare ceux-ci non convainquants car

reposant sur la subjectivité. "Their criteria and methods - little more that statements of belief - consist of a request for us to accept their judgments about character transformation, but we are note provided with a specific, logical basis an which to evaluate these judgments."
Il faut encore noter que cette attitude des cladistes par rapport au critère fonctionnel peut paraître aberrante à des non-cladistes. Ainsi Szalay (1981, p.38) par exemple accuse les cladistes à ce propos de négliger la signification biologique des caractères."... character analysis commonly practiced in cladistics is really a semantically enumeration of characters without attempts to analyze and weight them."

2.2.2. **Ordination des états.**

Ainsi donc, l'analyse des caractères se termine par la détermination de leur polarité. Actuellement, cette dernière se base essentiellement sur le critère ontogénique ("manuel") et surtout sur la méthode de l'outgroup (généralement programmée par ordinateur, après codage des caractères). Donc, il est bien clair, dès à présent, que l'utilisateur de programmes peut fournir à l'ordinateur des données dirigées ou non.

Avant de déterminer la direction de l'évolution des caractères, il convient de spécifier l'ordre de leur évolution. Lorsque les caractères sont binaires, cette étape est triviale. Dans le cas contraire, ce n'est pas toujours évident. Dans certaines circonstances, il peut exister un ordre logique entre les états (cf. Camin et Sokal, 1965 et Meacham, 1984). Ainsi, par exemple, on peut considérer, pour le caractère "couleur", que l'état "gris" est intermédiaire entre le "noir" et le "blanc". Pour les cas sans ordre logique, habituellement les séries de transformation sont obtenues, selon l'analyse de Mickevich (1982, p.461), à partir de théories plus générales ou "Règles de l'Evolution". Mais, fait-elle remarquer, ces règles, que ce soit dans le domaine morphologique ou moléculaire, sont actuellement très contestées. Ceci explique la remarque de Mickevich (1982, p.462) : "...the cladistic ordering of characters that have more than two states remains a significant and seemingly intractable problem."

Pour palier à ce problème, cet auteur (1982), propose une méthode nouvelle et très intéressante (cf. Coddington 1987), qu' elle appelle : Transformation Series Analysis (T.S.A.). Il s'agit d'une méthode itérative qui a pour but d'obtenir à la fois le cladogramme (voir paragraphe 3.1.2) le meilleur et les séries de transformation de caractères à plus de 2 états, dans le cadre d'une analyse de parcimonie de type Wagner (sensu stricto). Une de ses originalités consiste à tenir compte du cladogramme pour décider,à postériori, de l'ordre entre les états des

caractères. Mais dès lors une distinction supplémentaire s'impose : il faut considérer non seulement le caractère "original" ou initial, auquel correspond un arbre non orienté dont le code est fourni à titre d'hypothèse à priori à la T.S.A., mais aussi le caractère obtenu comme résultat de la T.S.A. et nommé caractère du cladogramme ("cladogram character") par Mickevich (1982). Une dichotomie analogue existe déjà chez Farris et al. (1970, p.184-185). Ces auteurs distinguent, en effet, d'une part, les arbres d'états de caractère primaire (ou à priori) qui sont l'input de l'analyse et d'autre part les arbres d'états de caractère secondaire (ou à postériori) qui sont fonction des résultats obtenus. Mais par rapport aux arbres à postériori de Farris et al. (1970), les caractères du cladogramme de Mickevich (1982) constituent une généralisation. En effet, ces derniers peuvent être "additifs" (c-à-d correspondre effectivement à un arbre), ou "non additifs" (lorsque le cladogramme ne présente pas d'indication de transformations entre paire d'états), ou "convergents" (si un état peut être obtenu à patir de plusieurs états différents) ou encore "disjoints"(quand il n'existe pas de transformation possible entre certains sous-ensembles d'états de caractère).
Mickevich (1982) a testé sa méthode sur différentes données de la littérature et a montré de fréquentes différences entre caractères originaux et caractères du cladogramme. Elle en conclut : "...'theories' for character evolution are not so well verified as has been presumed".

Il existe une autre possibilité de traitement des caractères à plus de 2 états : celle de faire l'hypothèse que, pour un caractère, un état peut se transformer directement en tout autre état. C'est sur cette base que repose l'optimisation de Fitch (1971) dont une variante est utilisée par Swofford (1985, p.3-7) dans l'option "unordered" de son programme PAUP. Cette approche est vivement recommandée, d'une manière générale par Swofford (1985, p.3-7) parce qu'elle évite toute hypothèse à priori de transformations.
Les caractères devant être codés pour correspondre à cette option sont ceux appelés "qualitatifs" par Day et Sankoff (1986) et "non-additifs" par Platnick (1988) et Farris dans son programme Hennig 86.
Selon Mickevich (1982) toutefois, cette méthode n'est valable que pour les séquences d'ADN, parce qu'effectivement une base peut être remplacée par toute autre base.

Comme en témoignent de violentes discussions entre Swofford et Mickevich au 7ème Congrès de la Société W. Hennig (1988), la controverse est actuellement très vive au sujet de la pertinence de l'utilisation en analyse de parcimonie des caractères "non ordonnés" avec optimisation de Fitch ou de caractères "originaux" avec T.S.A.

2.2.3. **Identification du caractère et détermination de ses états.**

Ainsi donc, l'utilisateur de programmes d'analyse cladistique peut actuellement éviter de fournir des données polarisées en laissant le soin à l'ordinateur d'enraciner l'arbre et il peut même se décharger, aux dépens de l'ordinateur, de l'ordination des états de caractères. Mais il y a une étape de l'analyse des caractères à laquelle il ne peut échapper : celle de l'identification d'un caractère et de ses états. Cette phase est particulièrement importante comme le souligne Meacham (1984, p.36) : ".. a careful evaluation of the ways in which similarities and differences among organisms are translated into characters for cladistic analysis will prove worthwhile."

Mais elle apparaît largement laissée à l'expertise du biologiste. Comme le remarque d'ailleurs Coddington (1987, p.181) : "The set of perceived 'states' seems to result from traditional perception of the comparative anatomy of the group being studied and that varies widely even between relativly closed groups." Il n'y a en effet dans la littérature cladistique que très peu d'information au sujet du choix des caractères et de la détermination de leurs états.

Les caractères doivent-ils être indépendants? Kluge (1989), fournit une réponse nuancée à cette question.
Il (1989 : 11) fait tout d'abord remarquer que l'hypothèse d'indépendance des observations existe dans toutes les sciences empiriques .
Il souligne ensuite que ,dans le contexte de la systématique phylogénétique, l'indépendance des caractères ne signifie pas la non-corrélation de ceux-ci. (Des caractères indépendants peuvent en effet être correlés en raison du partage d'un ancêtre commun.) En revanche, cette indépendance "semble impliquer que deux ... nouveautés ne peuvent être le résultat d'un même processus ... biologique."(Kluge, 1989 : 11). Cet auteur donne alors des exemples de mécanismes pouvant provoquer des changements évolutifs : il cite notamment des facteurs génétiques (tel l'association de gènes ou linkage) et des facteurs ontogéniques (comme la pédomorphose). Et il ajoute (1989:11) qu'il est extrêmement difficile de tester rigoureusement la covariance des caractères et que dès lors , dans l'école cladiste, on a tendance à ignorer l'hypothèse d'indépendance de ceux-ci. Kluge (1989 : 9), cependant, considère comme raisonnable de supposer qu' un ensemble de données morphologiques évolue séparément d'une collection d'observations biochimiques. Et il remarque que l'indépendance des données peut intervenir comme critère de pondération :" ...the phylogenetic hypothesis confirmed with the largest number of synapomorphies with the greatest likelihood of independance is preferred."(Kluge ,1989 : 11)

Faut-il , pour l'analyse cladistique, sélectionner certains caractères de préférence à d'autres ? En particulier, faut-il suivre la "tendance croissante à attribuer une importance plus grande aux données biochimiques" (Kluge,1989 : 12) qu'aux données morphologiques?
Selon Kluge (1989), il n'est absolument pas justifié d'agir ainsi. Ainsi donc , de la même façon que Gauthier et al. (1988) recommandent d'introduire ,dans l'inférence de cladogrammes, aussi bien les taxons fossiles qu'actuels (cf. parag. 1.2 de ce mémoire), Kluge conseille d'utiliser tous les caractères disponibles. Ceci est conforme au principe de l'emploi de la "totalité de l'évidence" ("total evidence") dont l'importance en systematique phylogénétique a récemment été mise en exergue par Kluge (1989).

2.3. CODAGE DES CARACTERES.

Comme signalé ci-dessus, le codage des caractères observés a pour but d'obtenir des variables numériques utilisées dans la matrice de données d'une analyse cladistique informatisée.

Le codage d'un caractère binaire est trivial : il utilisera une seule variable pouvant prendre des valeurs 0 et 1. Si ce caractère est de plus dirigé, alors l'état primitif de la variable recevra conventionnellement la valeur 0.

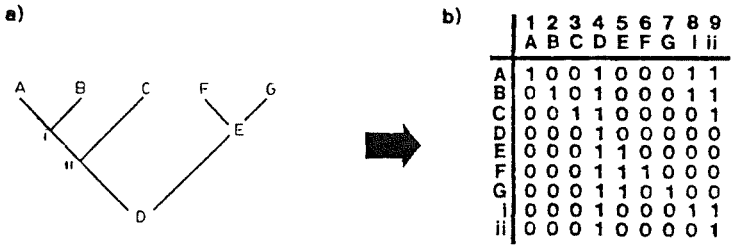
Quand le caractère est sans contrainte et cladistique, il doit être codé de telle façon que l'information relative à l'orientation et à la direction de ses états soit conservée. Le **codage additif binaire** répond à cette exigence. Il rend, de plus, les algorithmes plus aisés. Et enfin, malgré l'utilisation de beaucoup de variables, il permet des temps d'exécution très brefs (Goldman, 1989 :83). En effet , comme l'explique Swofford (1985 :4-3) : " ... tree lengths can be calculated using bit-counting operations and integer arithmetic ..." C'est pourquoi ce codage est fréquemment cité dans la littérature (Sokal et Sneath, 1963, Camin et Sokal, 1965; Farris et al. 1970, O'Grady et Deets, 1987 ...). Swofford (1985, p.4-3) utilise le codage additif binaire chaque fois que cela est possible dans PAUP et Felsenstein l'offre dans le programme FACTOR de PHYLIP réalisé par Meacham. Son principe sera donc détaillé ici. O'Grady et Deets (1987, p.269-271) en donne un exemple détaillé à partir duquel a été dérivée la règle plus formelle suivante :

---

* Le traitement de la valeur inconnue n'est pas, à ma connaissance, décrit dans la littérature et ne sera pas pris en compte ici.

1. A chaque état de l'arbre, faire correspondre une variable binaire.
   Ainsi dans la figure 2.2.b, la variable binaire identifiée par le chiffre 1 correspond à l'état A, la variable 2 à l'état B et ainsi de suite.

2. Dans une matrice carrée où les colonnes sont les variables binaires et les lignes, les états,
   donner à l'élément ij
   - la valeur 1 ssi l'état correspondant à la variable j se trouve sur le plus court chemin entre l'état i et la racine de l'arbre
   - la valeur 0 sinon.



	1 A	2 B	3 C	4 D	5 E	6 F	7 G	8 i	9 ii
A	1	0	0	1	0	0	0	1	1
B	0	1	0	1	0	0	0	1	1
C	0	0	1	1	0	0	0	0	1
D	0	0	0	1	0	0	0	0	0
E	0	0	0	1	1	0	0	0	0
F	0	0	0	1	1	1	0	0	0
G	0	0	0	1	1	0	1	0	0
i	0	0	0	1	0	0	0	1	1
ii	0	0	0	1	0	0	0	0	1

Fig. 2.2.[extraite de O'Grady et Deets 1987, p.270]
Codage additif binaire
a) L'arbre initial à 9 états de caractères à représenter dans une matrice numérique.
b) La matrice du codage avec 9 variables binaires.
(légende adaptée de O'Grady et Deets 1987 p.270)

Dans la figure 2.2.b, la ligne A contiendra la valeur 1 uniquement en colonne 1(A), 8(i), 9(ii) et 4(D) puisque le chemin le plus court entre A et la racine D passe par i et ii.
Comme vient de le faire remarquer Goldman (1989 : 80), la méthode proposée par O'Grady & Deets pour le codage binaire peut être simplifiée : cet auteur souligne en effet que la variable D de la fig. 2.2 est superflue car elle n'apporte aucune information. En conclusion, le nombre de variables binaires additives doit être égal ,au minimum, au nombre d'arêtes de l'arbre.

O'Grady et Deets (1987) et O'Grady et al. (1989) analysent d'autres méthodes de codage pour des caractères cladistiques à plus de 2 états : il s'agit du codage linéaire redondant ("Redundant linear coding"), du codage linéaire non-redondant

("Non redundant linear coding"), et de l'enracinement interne ("internal rooting"). Ces techniques ne seront pas expliquées ici. Par contre, les conclusions essentielles d'O'Grady et Deets (1987) à leur sujet seront mentionnées : ces deux auteurs montrent en effet que le codage linéaire redondant peut produire une pondération injustifiée de parties de l'arbre et doit donc être évité. Mais les deux dernières techniques sont pertinentes : le codage linéaire non-redondant est plus économique que l'additif binaire, en ce qui concerne le nombre de variables utilisées, et l'enracinement interne est approprié pour les arbres présentant une racine de degré 2. Tout récemment ,Goldman (1989) a proposé une méthode encore plus économe en variables que le codage linéaire non-redondant et appelée "fewest variables linear coding".
Quant au codage binaire non-additif (Farris et al. ,1970), il ne convient pas pour un caractère auquel correspond un arbre d'états. En effet, comme l'expliquent Pimmentel et Riggins (1987, p.204) avec ce codage : "... the form and order of the branched character are not conveyed. Moreover, this coding treats each state as an independent character ... Redundancy is introduced into the data and information content is sacrificed."

Il est clair que les codages valables pour les caractères ordonnés et non-dirigés le sont également pour ceux ordonnés mais non-dirigés.

CHAPITRE 3. INFERENCE D'ARBRES PHYLOGENETIQUES EN SYSTEMATIQUE PHYLOGENETIQUE.

Ayant déjà pris en compte les OTUs et les caractères de ceux-ci, il convient maintenant d'entrer dans le vif du sujet : l'inférence d'arbres phylogénétiques telle qu'elle est réalisée dans l'école cladistique ou phylogénétique. Cette analyse commencera par un retour aux sources.

3.1. SOURCES.

3.1.1. **Définitions préliminaires.**

Sauf spécification du contraire, les définitions qui vont suivre ont en effet été recherchées dans le texte original de Hennig (1966) lui-même. Toutefois la distinction entre caractère et état de caractère sera faite ici alors qu'elle est parfois omise chez Hennig.

Symplésiomorphie : partage d'états plésiomorphes d'un caractère. (traduit de Hennig 1966, p.89) (NB. Ceci est imputable à un ancêtre commun récent).

Synapomorphie : partage d'états apomorphes d'un caractère. (traduit de Hennig 1966, p.146-147) (NB. Ceci est un reliquat d'un ancêtre commun éloigné).

Autapomorphie pour un groupe monophylétique : possession exclusive d'un état dérivé de caractère par ce groupe (adapté de Hennig 1966, p.90).

Remarque : Si ce groupe contient plus d'un élément, l'autapomorphie pour le groupe est aussi la symplésiomorphie pour les éléments de ce groupe (Hennig 1966, p.90).

Convergence : obtention d'états similaires d' un caractère dans les espèces différentes sans que cela soit imputable à un ancêtre commun (adapté de Hennig 1966, p.117).
NB. Dans ce mémoire, "convergence" et "parallélisme" seront assimilés.

Réversion : ("Reversibility") passage d'un état dérivé à un état primitif pour un caractère (Hennig 1966, p.93).

Homoplasie : ce terme, inexistant chez Hennig, regroupe convergence et réversion.
Farris (1985a : 133) en donne la définition suivante : "Homoplasies ... are similarities that cannot be accounted for as resulting from inheritance; such similarities are not explained by a postulated genealogy, but are instead

CHAPITRE 3. INFERENCE D'ARBRES PHYLOGENETIQUES EN SYSTEMATIQUE PHYLOGENETIQUE.

Ayant déjà pris en compte les OTUs et les caractères de ceux-ci, il convient maintenant d'entrer dans le vif du sujet : l'inférence d'arbres phylogénétiques telle qu'elle est réalisée dans l'école cladistique ou phylogénétique. Cette analyse commencera par un retour aux sources.

3.1. SOURCES.

3.1.1. **Définitions préliminaires.**

Sauf spécification du contraire, les définitions qui vont suivre ont en effet été recherchées dans le texte original de Hennig (1966) lui-même. Toutefois la distinction entre caractère et état de caractère sera faite ici alors qu'elle est parfois omise chez Hennig.

Symplésiomorphie : partage d'états plésiomorphes d'un caractère. (traduit de Hennig 1966, p.89) **(NB. Ceci est un reliquat d'un ancêtre commun éloigné).**

Synapomorphie : partage d'états apomorphes d'un caractère. (traduit de Hennig 1966, p.146-147) **(NB. Ceci est imputable à un ancêtre commun récent).**

Autapomorphie Pour un groupe monophylétique : possession exclusive d'un état dérivé de caractère par ce groupe (adapté de Hennig 1966, p.90).

Remarque : Si ce groupe contient plus d'un élément, l'autapomorphie pour le groupe est aussi la symplésiomorphie pour les éléments de ce groupe (Hennig 1966, p.90).

Convergence : obtention d'états similaires d' un caractère dans les espèces différentes sans que cela soit imputable à un ancêtre commun (adapté de Hennig 1966, p.117).
NB. Dans ce mémoire, "convergence" et "parallélisme" seront assimilés.

Réversion : ("Reversibility") passage d'un état dérivé à un état primitif pour un caractère (Hennig 1966, p.93).

Homoplasie : ce terme, inexistant chez Hennig, regroupe convergence et réversion.
Farris (1985a : 133) en donne la définition suivante : "Homoplasies ... are similarities that cannot be accounted for as resulting from inheritance; such similarities are not

coincidental from the standpoint of the genealogical hypothesis."

Homologie : terme antinomique d'"homoplasie". A propos de l'homologie, Hennig (1966,p.94) reprend la définition de

Boyden (1947) :" ...nous définissons l'homologie comme toute similarité due à un ancêtre commun..." et il ajoute "Naturally in determining homologies we are limited to erecting hypotheses...".

Ces définitions seront complétées au paragraphe suivant à partir de la figure 3.1.

3.1.2. **Préceptes de Hennig.**

Les fondements de l'inférence d'arbres phylogénétiques (ou "phylogenetic kinship relationship" selon Hennig, 1966, p.150) peuvent être trouvés dans l'oeuvre de Hennig (1966).

En effet, trois principes de base y sont très explicitement formulés.

Le premier concerne la définition des séries de transformations (Hennig 1966, p.89) (cf.paragraphe 2.1). Hennig signale à ce propos qu'il se démarque de beaucoup de ses contemporains qui se basent sur le degré de correspondance morphologique ou encore sur la distinction entre caractères essentiels et non essentiels (Hennig 1966, p.146).

Le deuxième principe, absolument fondamental, stipule que ce sont uniquement les synapomorphies (et non les symplésiomorphies) qui justifient la présomption de monophylie pour un groupe (Hennig 1966, p.93 et 90). A ce sujet, il convient de mentionner qu'Hennig (1966, p.93) ajoute modestement : "Many authors call the recognition that only synapomorphy, not symplesiomorphy, can be the basis for monophyletic groups 'commonplace'." Ceci est très rarement mentionné dans la littérature ! A noter encore qu'une formulation plus précise de ce deuxième principe peut être trouvée chez Farris et al. (1970, p.173 et 174).

Le troisième principe peut se traduire de la façon suivante : plus un groupe présente de synapomorphies, mieux est fondée l'hypothèse de monophylie à son égard (Hennig 1966, p.91).

Enfin, ceux qui, à la suite de Farris et al. 1970, analysent dans le détail le livre de Hennig 1966, découvriront, isolé des précédents, un dernier principe qualifié d'"auxiliaire" par Hennig lui-même (1966, p.121). Une adaptation de la formulation de ce principe par Farris et al. (1970, p.175) pourrait être : en l'absence d'évidence du contraire, tout état dérivé partagé par les membres d'un groupe doit être considéré comme ayant émergé une seule fois dans ce groupe.

Si Hennig formule des principes de bases, il omet par contre d'expliquer formellement la démarche à suivre pour construire un

arbre phylogénétique. Il se contente d'annoncer laconiquement : "All these consideration are summuarized in a 'scheme of argumentation of phylogenetic systematics' " (Hennig 1966, p.90) et de produire la figure ci-dessous.

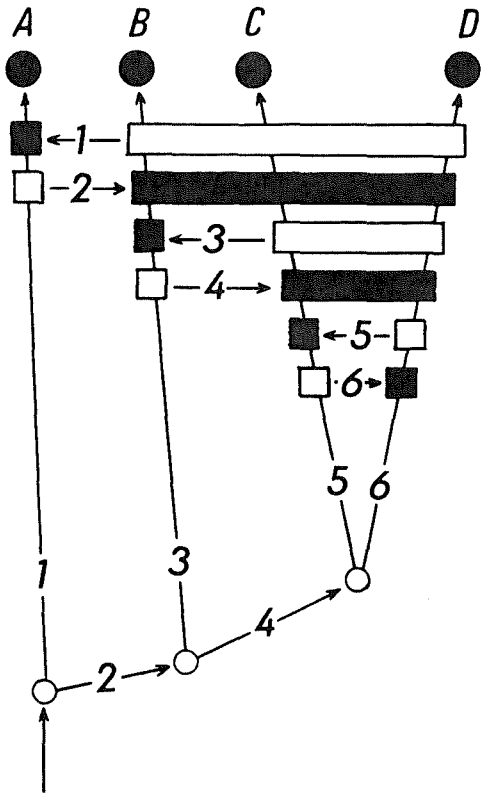


Fig.3.1.(extraite de Hennig 1966, p.91) "Schéma d'argumentation de la systématique phylogénétique".
Légende :adaptée du texte (p.90 et 91) de Hennig.
* pour les 6 caractères binaires numérotés de 1 à 6 :
un carré noir = un état apomorphe
un rectangle noir = une synapomorphie
un carré blanc = un état plésiomorphe
un rectangle noir = une symplésiomorphie
* Les taxons sont symbolisés par des disques et éventuellement identifiés par une lettre.
* Les flèches indiquent le sens des transformations des états de caractères (Hennig 1966, p.91) et aussi une relation de filiation entre taxons.

Et il fait le commentaire suivant : Pour des raisons de simplicité, une seule autapomorphie est supposée pour chaque groupe monophylétique (A, B, C, D et les groupes reliés par des rectangles B+C+D et C+D) (Hennig 1966, p.90).

La figure ci-dessus représente un arbre phylogénétique selon Hennig qui sera, postérieurement à cet auteur, qualifié de enraciné (au sens cladistique) (cf.paragraphe 3.2.0.) et encore appelé cladogramme.
Dans un tel arbre s'expriment des relations de parentés sous

forme de mise en évidence de groupes frères (ou "sister groups" selon Hennig) ayant l'exclusivité d'un ancêtre commun.Deux taxons apparaissent comme "sister group" s'ils sont plus étroitement liés entre eux qu'à tout autre taxon (Engelmann et Wiley (1977,p.6)). Comme le formule joliment Tassy (1986) pour traduire l'expression "recency of common ancestry" utilisé par Hennig (1966, p.72), un cladogramme s'explique donc "en termes de degrés d'ascendance commune". Il faut bien noter que dans un tel arbre, les taxons observés sont obligatoirement des feuilles et les ancêtres hypothétiques, des noeuds. Une distinction analogue n'existe pas dans un arbre d'états de caractères (cf.paragraphe 2.1)!

Hennig lui-même fait remarquer un peu plus loin (1966 : 93) que son exemple est essentiellement simplifié dans la mesure où il n'y a pas de conflits de caractères : l'arbre obtenu est absolument dépourvu de réversion et de convergence. Dès lors, en utilisant les principes de Hennig, un seul cladogramme s'impose. Mais Hennig n'exprime pas explicitement l'attitude à adopter en cas d'homoplasie, lorsque ses 4 principes sont insuffisants et qu' il faut choisir entre plusieurs cladogrammes possibles. C'est pourquoi différentes méthodes utilisant un critère d'optimisation suffisamment précis se sont développées : un premier groupe très diversifié relève de la parcimonie et un second, d'importance moindre, de la compatibilité (voir les définitions au 3.2.1.1.).

3.1.3. **Hennig a-t-il préconisé la parcimonie** ?

La réponse à cette question varie selon les auteurs.

Wiley (1981, p.176) l'affirme ("... like Hennig's method, they arbitrate between conflicting hypotheses using parsimony.") mais il n'en fournit aucune preuve.

Selon Farris, il existe un lien très étroit entre la méthode préconisée par Hennig (1966) et l'analyse de parcimonie. Une très belle démonstration formelle peut en être trouvée dans un article déjà ancien (Farris et al.1970). Dans ce dernier, les principes dégagés ci-dessus de l'ouvrage de Hennig sont clairement énoncés, voire même précisés sous forme d'axiomes. Ensuite, Farris et al. (1970) utilisent ces axiomes pour démontrer des théorèmes relatifs à la construction d'un arbre phylogénétique, aussi bien dans les cas où les réversions sont interdites (cas Camin-Sokal) que si elles sont permises (cas Wagner sensu stricto avec arbre non-enraciné).
De plus, les germes d'un critère d'optimisation pour choisir entre différents arbres possibles sont déjà soulignés par Farris et al. (1970) dans le troisième principe de Hennig. Les auteurs font toutefois remarquer que ce critère n'est pas suffisamment détaillé car il ne permet pas d'évaluer un arbre composé de plusieurs groupes monophylétiques. Dans ce cas là, le critère de parcimonie doit être utilisé, selon Farris et al., mais il n'est pas en contradiction avec le troisième principe de Hennig car "...on a most parsimonious tree, OTUs that share many steps (...) are generally placed together. We might argue that the parsimony

criterion selects a tree ... by 'averaging' in some sense the preferability of all the monophyletic groups of the tree" (Farris et al. 1970, p.176).

S'opposant à ces auteurs, Felsenstein (1983, p.316) professe que la parcimonie ne s'enracine pas davantage dans l'oeuvre de Hennig qu'une autre méthode de résolution de conflits parmi les caractères. Et il juge tendancieuse l'argumentation de Farris et al. citée ci-dessus: "their argument contains one step that leads to the arbitrary selection of parsimony...".(Felsenstein ,1983).

Quant à Duncan (1984), il professe que c'est "l'analyse de compatibilité qui est équivalente à la méthode de Hennig".Selon lui, en effet, Hennig restreint la signification du terme synapomorphie "au partage d'apomorphies qui sont supposées être apparues une seule fois".

A deux reprises récemment, Farris (1985 et Farris et Kluge 1986) fait l'apologie de la parcimonie en tant que méthode préconisée par Hennig et s'en prend à la position de Duncan (1984) .Selon lui, Duncan néglige de prendre en considération de nombreux exemples donnés par Hennig ultérieurement (1983) et à partir desquels l'acception réservée par cet auteur au mot "synapomorphie" peut être clairement déduite. D'après Farris, cette acception ne coïncide absolument pas avec celle défendue par Duncan.

3.2. UTILISATION D'UN CRITERE D'OPTIMISATION SUFFISANT POUR RESOUDRE LES CONFLITS DE CARACTERE.

3.2.0. **Définitions préliminaires.**

Racine: (au sens cladistique): ancêtre commun à tous les OTUs analysés. Selon la définition de Farris (1980,p.83):" ...the root is presumed to represent a point chronologically prior to any descendent point.".

N.B.: La racine (au sens cladistique) doit être distinguée de la racine d'un arbre binaire au sens de la théorie des graphes. (cf. chapitre 6.).
Au niveau de la racine (cladistique),l'état de chaque caractère est, par définition même, primitif.

Arbre phylogénétique enraciné: présentant une racine (au sens cladistique).

N.B.: Un cladogramme (cf. Camin et Sokal, 1965 : 312) est un arbre phylogénétique enraciné.

Arbre phylogénétique non-enraciné : sans racine (au sens cladistique). Comme le souligne Swofford (1985,p.37), un tel arbre implique le même nombre de changements évolutifs quelle que soit la position de la racine (cladistique)

N.B.: A partir d'un arbre non-enraciné cladistiquement, on peut obtenir beaucoup d'arbres cladistiquement enracinés différents selon la localisation choisie pour situer la racine cladistique.
Cette dernière peut coïncider avec un sommet existant de l'arbre non-enraciné cladistiquement [cf. ROOT=ANCESTOR dans le programme PAUP de Swofford (voir chapitre 7) et cf. Farris (1970 , p.84)].

Mais elle peut aussi constituer un sommet supplémentaire venant s'inscrire sur une arête de l'arbre cladistiquement non-enraciné et la diviser en deux (voir figure 3.2). Ceci est typiquement ce qui se passe lors de l'enracinement par outgroup (ou extra-groupe) tel qu'il est réalisé par MIX (cf. chapitre 6) et aussi par PAUP (cf. chapitre 7).C'est d'ailleurs ce que décrivent Colles (1985 : 315) et Felsenstein (1983 : 315) : "If we were to specify that species D was an outgroup, so that the root must lie on the line connecting D to the rest of the tree ..."

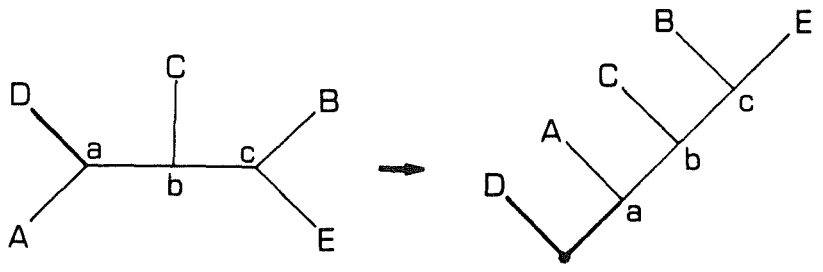


Fig. 3.2 : Enracinement par outgroup (constitué d'un OTU unique) (Illustration du choix de placer la racine cladistique sur l'arête Da de l'arbre cladistiquement non-enraciné c.à d. de désigner l' OTU D comme outgroup.)
N.B.: L'arbre enraciné (cladistiquement) est ici dessiné comme c'est généralement l'usage :sans flèches directionnelles mais avec la racine cladistique occupant une position basale par rapport à l'arbre.

Si l'outgroup comprend plusieurs OTUs (possibilité offerte par PAUP) , alors la racine viendra séparer 2 noeuds de l'arbre non-enraciné.Il est fondamental de remarquer que la racine cladistique représente, dans ce cas aussi, l'anc@tre commun à la fois de l'outgroup et du reste des OTUs analysés.[Ceci est conforme à la définition de la racine selon Farris (1970),citée ci-dessus.]

3.2.1.**Présentation non formelle.**

Avant de présenter formellement le problème général d'optimisation de l'inférence d'arbres phylogénétiques intervenant en analyse cladistique numérique pour des variables discrètes (cf. paragraphe 3.2.2), il sera fait un exposé non formel et plus facilement accessible de différentes instanciations de celui-ci (c-à-d de problèmes particuliers). Cette première présentation se veut très générale. Mais en relation avec celle-ci, des formulations mathématiques, et un complément de références bibliographiques peuvent @tre trouvés dans les articles suivants : Day, 1983; Day et Sankoff, 1986; Day, Johnson et Sankoff, 1986 et Day et Sankoff, 1987).

La classification des différents problèmes numériques retenus ici (cf. tableau 3.1) est celle qui peut précisément @tre déduite des textes dont les références viennent d'@tre citées. Cette classification peut aider celui qui aborde la littérature cladistique. Une analyse légèrement différente peut @tre trouvée chez Felsenstein (1982 et surtout 1983).

3.2.1.1 Classification des différentes instanciations du problème.

3.2.1.1.1 Parcimonie.

Selon Felsenstein (1983), les premiers à avoir utilisé le critère de parcimonie dans un contexte phylogénétique (-tout en l'appelant "the method of minimum evolution-) sont les généticiens des populations Edwards et Cavalli-Sforza (1963).

C'est sous leur expression numérique et informatisée que les méthodes de parcimonie connaissent leur plein épanouissement. Mais il en existe une version manuelle, qualifiée de "classique" par Wiley (1981) ou encore de "traditionnelle" qui sera brièvement évoquée ici à la fois à propos du critère d'optimisation retenu et des solutions proposées.

3.2.1.1.1.1 Parcimonie manuelle ou classique.

Le principe de parcimonie (ou de simplicité ou d'économie) est défini de manière informelle par Wiley (1981, p.20) comme celui qui est utilisé pour choisir l'hypothèse c-à-d. l'arbre phylogénétique qui explique les données de manière la plus économique. En vertu de ce principe, il faut minimiser les "hypothèses ad hoc" d'homoplasie*. Il est intéressant de souligner que le recours au principe de parcimonie tel que défini ci-dessus, s'inscrit dans le cadre d'une démarche effectuée, il y a quelques années, par certains cladistes (Miles, 1973; Gaffney, 1979; Wiley, 1981 ...) de se référer à la philosophie des sciences de Popper (1968) pour souligner le caractère scientifique de leurs méthodes.

Concrètement , Wiley (1981, p.140 à 141) montre, à partir d'un exemple, comment il construit manuellement en prenant en compte successivement chacun des caractères, un arbre phylogénétique d'emblée enraciné, basé sur le principe de parcimonie et le critère de l'outgroup.

Cette approche classique connaît encore de nombreux adeptes actuellement (cf. par exemple Andrews et Martin, 1987 ou encore Sæther,1986). Mais elle est contestée. En effet, comme le souligne Pimentel et Riggins (1987, p.202), une erreur, même minime, réalisée au départ, peut aller en s'amplifiant de telle façon que le cladogramme résultant ne soit pas en accord avec les données. Les deux auteurs préconisent en revanche une démarche où le choix du meilleur arbre relève d'une seule décision basée sur la totalité des relations entre les états de tous les caractères. Cette critique de Pimentel et Riggins me paraît aggréger deux accusations différentes. La première qui serait de souligner que la démarche manuelle est non exhaustive et que l'heuristique qu'elle contient peut être insuffisante. La seconde, qui serait de lui préférer une méthode exacte construisant successivement tous les arbres possibles avant de choisir le meilleur. Mais ici une objection majeure me paraît alors s'imposer : il est très difficile d'envisager mentalement toutes les possibilités même si le nombre d'OTUs et d'états de caractères est petit. La preuve en est que, pour l'exemple pourtant très simple proposé par Wiley (1981), Maddison et al. 1984, p.94) proposent une autre solution tout aussi parcimonieuse.

A noter encore que la méthode de Wiley est relativement "standard" mais il en existe des variantes (cf. par exemple Skelton et al. 1986).

---

* Cette assimilation des hypothèses ad hoc et des homoplasies est contestée par Forster (1986).

Quoi qu'il en soit, une méthode manuelle ne peut être envisagée que si le volume des données est faible. C'est le cas par exemple de l'analyse cladistique de certaines araignées à partir de leur comportement (Coddington, 1986).

3.2.1.1.1.2. Parcimonie numérique .

La traduction numérique ou quantitative du critère de parcimonie est de sélectionner l'arbre phylogénétique minimal (voir vocabulaire relatif à la théorie des graphes), quelle que soit d'ailleurs la façon dont la longueur de ses arêtes est évaluée. C'est sur cette formulation du critère que sont basés les programmes d'informatiques.

Fréquemment, les cladistes désigneront l'arbre "minimal" par les termes "le plus parcimonieux". De la même façon, ils se réfèreront au "critère de parcimonie à maximiser".

Le problème de Steiner est cité ici pro forma. Il s'applique aux séquences d'acides aminés dans les protéines. Chaque OTU y est représenté par un t-tuple de symboles nominaux identifiant les acides aminés (Les variables sont donc ici discrètes mais non numériques).

Tableau 3.1 Classification de différentes instantiations du problème de l'inférence d'arbres phylogénétiques à partir de variables discrètes en analyse cladistique numérique.

1. PARCIMONIE

[1.1. Problème de Steiner]

1.2. Construction d'arbres de Wagner
- 1.2.1. type Wagner sensu stricto
- 1.2.2. type Camin -Sokal
- 1.2.3. type Dollo
- 1.2.4. type polymorphisme ou inversion de chromosome

2. COMPATIBILITE

Les arbres de Wagner (cf. Wagner 1961) seront davantage détaillés car ils occupent une position fondamentale en analyse phylogénétique. La longueur de leurs arêtes est mesurée dans la métrique de Manhattan (encore appelée "City block") (cf. Day, 1983 : 434.) Cette dernière est définie de la façon suivante :

soit.X : une matrice de variables réelles.
.A,B : 2 sommets.
.X(A,i) : l'état pour le sommet A de la variable i dans X

$$d(A,B) = \sum_i |X(A,i) - X(B,i)|$$

Cette fonction d est aussi appelée différence (Farris, 1970) ou encore différence phénétique (Farris, 1972) entre 2 sommets.

L'utilisation conjointe de la métrique de Manhattan (de préférence à la métrique Euclidienne) et de variables discrètes qui caractérise les arbres de Wagner, permet d'interpréter la longueur de l'arbre comme la somme des changements évolutifs qu'il implique.
Et dès lors, un arbre minimal est aussi un arbre pour lequel le nombre de réversions et de convergences est minimisé (cf. parcimonie classique).

Il existe différentes variantes possibles d'arbres de Wagner en phylogénétique systématique.
Si les réversions et les convergences sont autorisées,on parlera, dans ce mémoire,du type Wagner (sensu stricto). Pour ce dernier, la formalisation mathématique est due à Kluge et Farris (1969) et surtout à Farris (1970). Un arbre phylogénétique de type Wagner sensu stricto présente la particularité remarquable de pouvoir être non-enraciné cladistiquement.
Si les convergences sont permises à l'exclusion des réversions, (cf.fig.3.3.a) il s'agit du type Camin-Sokal (cf. Camin et Sokal, 1965 et Estabrook, 1968). Ces 2 premiers types sont très utilisés pour des données morphologiques.
Si au contraire, seules les convergences sont interdites (cf.fig.3.3.b), on obtient le type Dollo (cf. Le Quesne, 1975 et 1977 et Farris 1977 a et b). Ce type est ainsi nommé parce qu'il s'inspire de la loi dite de Dollo (1893), selon laquelle une structure complexe est rarement acquise mais facilement perdue au cours de l'évolution. Ce type de parcimonie semble particulièrement appropriée pour les sites de reconnaissance d'enzyme de restriction dans l'ADN (voir par exemple à ce sujet l'étude de l'ADN mitochondrial animal réalisé par Debry et Slade, 1985).
Un dernier type de parcimonie est nommé inversion de chromosome par Farris (1978) et polymorphisme par Felsenstein (1979) (cf.fig.3.3.c.). Ce modèle est tout à fait particulier puisqu'il permet la coexistence de l'état dérivé et de l'état primitif (c.-à-d.le polymorphisme). Cette dernière ne peut surgir qu'une seule fois au cours de l'évolution, mais elle peut subir des réversions vers l'état primitif ou vers l'état dérivé (cf. Felsenstein 1983,

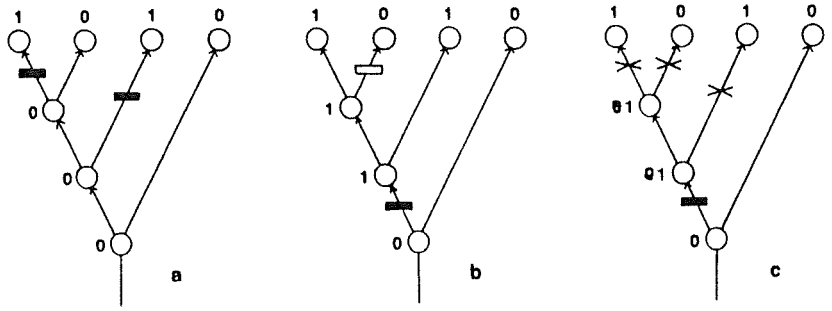


Fig. 3.3 : Arbres de Wagner: type Camin-Sokal (a), type Dollo (b) et type polymorphisme (c).
Les changements 0->1 (ou 0->01) sont symbolisés par des rectangles noirs, les réversions de 1->0 par des rectangles blancs et les pertes du polymorphisme par une croix.
(figure extraite et légende adaptée de Felsenstein 1983).

p.319). Ce type de parcimonie peut être utilisé à propos d'inversions de chromosomes (cf. Farris 1978). En effet, comme le souligne Day et Sankoff (1987, p.214) à ce propos, on peut en effet être réticent à envisager le même type d'inversion se produisant plus d'une fois au cours de l'évolution au même site et on peut préférer expliquer l'apparente convergence dans une phylogénie par la persistance de la condition hétérozygote (c-à-d polymorphe dans le vocabulaire de Felsenstein).

Les types d'arbres de Wagner décrits ci-dessus sont "purs",c-à-d qu'ils concernent des modèles où tous les caractères (/ variables)subissent les mêmes contraintes.
Mais des arbres "mixtes", c.-à-d. où les caractères eux-mêmes (et donc aussi les variables correspondantes) relèvent de types de parcimonie (encore appelés ici types évolutifs) différents, peuvent aussi être envisagés. C'est ainsi par exemple que le programme MIX, qui sera analysé plus loin, permet de construire des arbres de Wagner où certains caractères (/ variables)sont de type Camin-Sokal et d'autres de type Wagner (sensu stricto).

Se trouvent encore classés par Day (1983) dans les problèmes de parcimonie, la construction d'arbres additifs évolutifs. Ces derniers se distinguent des arbres de Wagner par le fait qu'ils sont établis directement à partir de matrices de distances entre paires d'OTUs.Cette particularité leur supprime la possibilité d'être considéré comme une instanciation du problème général tel qu'il sera défini au 3.2.2.2. Les arbres additifs sont fréquemment utilisés lorsque les valeurs des caractères des OTUs sont inconnues, et que dès lors l'analyse des caractères telle qu'elle est décrite ci-dessus n'est pas pertinente. C'est le cas par exemple pour les données relatives aux distances moléculaires (cf. distances immunologiques ou

d'hybridation de l'ADN) entre taxons.
Les arbres additifs constituent un domaine important en analyse cladistique, mais ils font l'objet de controverses (Farris, 1986a et Felsenstein, 1986).

3.2.1.1.2 Compatibilité.

L'analyse de compatibilité constitue une entité bien particulière et utilise un vocabulaire propre. Une synthèse de l'évolution historique de celle-ci est donnée par Felsenstein (1982, p.389-393) et une bonne bibliographie par Meacham (1981). Seules quelques définitions seront fournies ici.

Des caractères mutuellement compatibles sont ceux pouvant s'inscrire sans réversion ni convergence sur un seul et même arbre phylogénétique (Felsenstein, 1981, p.390). Une clique est un ensemble de caractères mutuellement compatibles (Meacham, 1981, p.592). Les techniques de compatibilité sont celles dans lesquelles **la plus grande clique est recherchée** (Meacham, 1981).

3.2.1.1.3 Relations logiques entre ces différentes instanciations du problème.

Un lien conceptuel entre compatibilité et parcimonie est souligné par Felsenstein (1983, p.319) : en maximisant le nombre de caractères non-homoplasiques (cf. compatibilité), on minimise, de facto, le nombre de caractères homoplasiques. Or ceci est nettement apparenté à la minimisation du nombre de changements évolutifs (parcimonie). Le Quesne (1982, p.274) considère la compatibilité et la parcimonie comme complémentaires et préconise d'ailleurs l'usage de ces deux techniques pour inférer des phylogénies.

Il faut encore noter que Felsenstein (1979), à partir d'un seul modèle probabiliste de l'évolution d'un caractère, a montré qu'il pouvait faire émerger comme méthodes de maximum de vraisemblance, 4 méthodes différentes d'inférence phylogénétique : la parcimonie de type Dollo, de type Camin Sokal et de type polymorphisme ainsi que la compatibilité. Ceci souligne selon cet auteur le lien logique existant entre ces différentes méthodes.

3.2.1.2 Nature des problèmes et conséquences.

3.2.1.2.1 Définition des problèmes NP-complets.

"Lorsqu'on examine un algorithme A, on trouve presque toujours un paramètre, soit p, caractérisant la taille des données auxquelles A s'applique dans chaque cas. Soit donc T(p), le temps d'exécution de l'algorithme A, exprimé en fonction de p. On dit que l'algorithme A est de complexité théorique O (f(p)), où f est une certaine fonction du paramètre p, si la croissance asymptotique de T(p) est de l'ordre de f(p) au plus ou, autrement dit, si le rapport T(p)/f(p) est borné lorsque p tend vers + ∞, ou encore s'il existe une constante c telle que T(p) ≤ cf(p) pour toute valeur positive de p, sauf peut-être pour un ensemble fini de valeurs positives de p. C'est le cas notamment lorsque T(p)/f(p) tend vers une constante c pour p tendant vers + ∞." (Fichefet, I.85)

"L'idée est que l'on peut mesurer la complexité d'un problème par la complexité du meilleur algorithme qui le résout. On dit qu'un algorithme est polynômial si son exécution demande un nombre minimum d'opérations, borné par une fonction polynômiale de la taille des données d'entrée. Le problème correspondant est alors qualifié de traitable. Entre les problèmes traitables et les autres (i.e. ceux pour lesquels on sait démontrer qu'aucun algorithme polynômial ne peut produire une solution), il y a une vaste classe de problèmes. Parmi ceux-ci, on distingue les problèmes NP*-complets. On ne connaît aucune algorithme polynômial pour les résoudre, mais on ne sait pas, non plus, démontrer qu'ils ne sont pas traitables. On sait seulement depuis Cook (1971) que si un seul d'entre eux était reconnu polynômial, tous les autres problèmes NP-complets le deviendraient" (Barthélemy et Guénoche, 1988, p.129).
D'autre part, "ces problèmes ont une propriété très intéressante. Dès qu'une solution est trouvée pour un exemple particulier du problème (...), il est possible de vérifier que la solution est correcte. Plus précisément, pour chacun de ces problèmes il existe un algorithme qui, étant donné un exemple particulier du problème et une solution proposée, peut vérifier en un temps polynomial si la solution est correcte ou si elle ne l'est pas." (Goldschlager et Lister, 1986 : 115)
Une définition formelle de ces problèmes peut être trouvée dans Mandrioli et Ghezzi (1987 : 256) et une très belle approche de vulgarisation , dans Lewis et Papadimitriou (1978).
Comme le signalent Graham et Foulds (1982, p.137), quelques uns des meilleurs mathématiciens ,depuis une trentaine d'années ont vainement essayé de trouver une solution polynômiale à ces problèmes.

---

* NP- pour "Nondeterministic Polynomial" c.-à-d. "polynomial non déterminé" (cf. Lewis et al. 1978 : p 69).

Et Barthélemy et Guénoche (1988, p.130) de conclure : "Le point important est que, compte-tenu des efforts qui ont été déployés dnas cette direction, il est très peu probable que quiconque arrive à exhiber un algorithme polynômial pour résoudre un problème NP-complet!".

3.2.1.2.2 Application de cette définition aux problèmes d'inférence d'arbres phylogénétiques en analyse cladistique.

Il est actuellement démontré que tous les problèmes d'optimisation intervenant dans l'analyse cladistique qui ont été mentionnés au paragraphe précédent, sont NP-complets. En effet, tombent dans cette catégorie de problèmes aussi bien celui de Steiner (cf. Foulds et Graham, 1982 et Graham et Foulds, 1982) que celui de construire un arbre de Wagner, quel que soit son type (Day, 1983; Day, Johnson et Sankoff, 1986; et Day et Sankoff, 1986) ou un arbre évolutif additif (Cf. Day, 1983) ou encore d'inférer une phylogenèse par compatibilité (Day et Sankoff, 1986).
Et ceci a été prouvé pour tous les types possibles de caractères fournis comme données, c.-à-d. binaires ou sans contrainte, cladistiques ou qualitatifs aussi bien pour les arbres de Wagner de types Camin-Sokal et Dollo (Day, Johnson et Sankoff, 1986) que pour les arbres obtenus par compatibilité (Cf. Day et Sankoff, 1986). Et c'est encore vérifié, tant pour les caractères cladistiques que qualitatifs, dans les cas de la parcimonie de type polymorphisme (Day et Sankoff, 1986).
Rohlf (1984) illustre, à l'aide d'un exemple très didactique, une des raisons pour laquelle le problème de trouver l'arbre de Wagner le plus court est tellement difficile : il montre très bien que la longueur de l'arbre n'est pas fonction uniquement des relations "locales" entre les OTUs. "To achieve the globaly minimum length, all distances must be taken into consideration simultaneously ..." (Rholf, 1984, p.342).

3.2.1.2.3 Conséquences relatives aux algorithmes.

Il existe, en systématique phylogénétique, des algorithmes exacts c.-à-d. garantissant de fournir la solution optimale.

Les uns construisent exhaustivement tous les arbres pour sélectionner le meilleur. Mais le nombre d'arbres différents possibles augmente très rapidement avec l'accroissement du nombre d'OTUs et/ou du nombre d'états de caractères à prendre en compte.

Cavalli-Sforza et Edwards (1967) ont calculé qu'à partir de n OTUs , on peut construire $(2n-3)![2^{n-2}(n-2)!]$ topologies d'arbres binaires dont ces OTUs sont les feuilles. Ainsi par exemple, à 9 espèces observées correspondent 2027025 cladogrammes possibles (Mickevich et Platnick, 1989 : 43) !

D'autres utilisent la technique du "branch and-bound" (cf. par exemple Hendy et Penny, 1982). Mais l'utilisation de ces stratégies doit toujours être limité à des données restreintes : une douzaine d'OTUs au maximum dans la version 3.0 de PAUP de Swofford.

Pour les problèmes avec un volume de données important, il est donc justifié de développer des algorithmes heuristiques (c.-à-d. ne garantissant pas de fournir les arbres optimaux mais ne consommant qu'un temps d'ordinateur raisonnable) plutôt que d'essayer d'obtenir des algorithmes exacts. C'est ce que conclurent Graham et Foulds de leur étude (1982, p.137 : "... the fact that a problem is NP-complete is considered justification for heuristic procedures to be applied... The challenge is to find heuristics with good performance guarantees which [also have efficient algorithms]."

Day (1986, p.228) fait d'autre part remarquer que certains algorithmes , même s'ils requièrent un temps d'exécution exponentiel dans le pire des cas, peuvent cependant être relativement efficients pour des ensembles de données incluant ceux rencontrés dans la pratique. Il en déduit qu'il est intéressant aussi de chercher à caractériser des classes restreintes de problèmes pour lesquels les algorithmes exacts sont efficients (c.-à-d. de temps polynômial) (Cf. Day et Sankoff, 1987, p.218).

3.2.2. **Présentation formelle du problème général de l'inférence d'arbres phylogénétiques en analyse cladistique numérique avec variables discrètes dans le cadre d'une comparaison avec les langages formels.**

Ce paragraphe aboutira à la formulation mathématique contenue dans le tableau 3.4. Cette formulation présente la double particularité de regrouper en une seule définition générale les problèmes d'optimisation répertoriés au tableau 3.1 et d'utiliser une terminologie empruntée aux grammaires formelles.

Au préalable cette terminologie sera précisée et , à cette occasion , sera réalisée une comparaison entre un "système cladistique" et différentes structures formelles.

3.2.2.1 Essai de mise en parallèle de l'inférence d'un arbre phylogénétique et de la génération d'un langage formel.

3.2.2.1.1. Introduction.

C'est à Monsieur Barreto que revient le mérite d'avoir perçu une possibilité de rapprochement entre ces deux domaines et d'en avoir suggéré une étude. Brooks et Wiley (1985) comparent également les grammaires génératives et l'analyse cladistique. Mais leur approche est totalement différente. En effet, d'une part, elle se situe à un autre niveau : essentiellement ,elle souligne la dichotomie existant dans les langages formels entre la structure profonde (ou sémantique) et la structure superficielle (ou phonétique).
D'autre part, elle a pour but d'expliquer la différence entre "pattern cladist" et "phylogenetist cladist" comme relative à une dichotomie analogue à celle rencontrée dans les grammaires génératives: "For one group, the emphasis is on surface structure; for the other emphasis on what surface structure can tell us about deep structure". (Brooks et Wiley, 1985, p.4).

Au préalable, il est éclairant de rappeler ici que la <u>généralisation</u> d'un concept se réalise en supprimant des parties de la définition de manière à pouvoir inclure de nouveaux objets, tandis que sa <u>particularisation</u> implique au moins une ajoute.

Par souci de concision, le texte ci-dessous suppose connues les notations ensemblistes et celles de la théorie des langages formels.(cf. par exemple Hopcroft et Ullman,1968)

A titre exemplatif, il est illustré par un cas de figure, voisin de celui du programme MIX (3.1) de Felsenstein qui sera longuement analysé dans la deuxième partie de ce mémoire. MIX infère un arbre de Wagner à partir de variables pouvant valoir 0 ou 1 et être de type Wagner sensu-stricto ou de type Camin-Sokal. D'autre part, ce programme est capable de manipuler la valeur

inconnue (?) comme état de variable (voir paragraphe 6.5.2.). Ce "?" signifie en réalité que l'état peut être aussi bien 0 que 1 . (L'état polymorphe, voir paragraphe 6.5.3., n'est pas pris en compte ici). L'exemple utilisé ici restera à un niveau plus conceptuel que MIX : il sera concerné par les caractères eux-mêmes (et non les variables).
D'autre part, le cadre de l'analyse cladistique est, dans l'exemple choisi, plus général que celui de MIX : ici l'arbre à construire n'est pas binaire mais ternaire. Ainsi donc, à supposer qu'il y ait n OTUs à analyser plus la racine cladistique, l'arbre correspondant comportera 2(n+1)-2 sommets.
De plus, on impose que la racine cladistique soit de degré égal à un. [Il faut remarquer que ces deux dernières clauses sont exactement celles de l'option ROOT=ANCESTOR du programme PAUP (2.4) de Swofford (cf.paragraphe 7.2.3.).]

3.2.2.1.2.Système Cladistique - Langage généré - Reconnaissance de mots .

a) Appelons **"Système Cladistique"** SC la construction suivante : $\langle V_c , P , \omega \rangle$ où

V_c est le vocabulaire cladistique
soit dans l'exemple ci-dessus : {1,0,?} .

P appelé table de SC,
est une relation finie non-vide telle que :
$P \subseteq V_c^{*} \times V_c^{*}$ (où V_c^{*} = ensemble de tous les mots composés de symboles de V_c)
et, pour chaque $\langle \alpha, a, \beta \rangle$ dans $V_c^{*} \times V_c \times V_c^{*}$, il existe un γ dans V_c^{*}
tel que $\langle \alpha, a, \beta, \gamma \rangle \in P$.
(Chaque élément de P est appelé une production ou règle de réécriture ou encore une transformation)
L'ensemble des règles de réécriture satisfait à une structure de semi-groupe.

ω est l'axiome, c-à-d. un mot composé de symboles du vocabulaire V_c . Dans tout système cladistique, l'axiome peut être assimilé à la racine cladistique.

Pour chacun des éléments de cette définition, des commentaires et des explicationss s'imposent.

Par rapport à V_c tout d'abord, il faut signaler que l'existence d'un vocabulaire unique singularise un SC par rapport à une grammaire formelle standard telle qu'elle est définie par exemple dans Hopcroft et Ullman (1969, p.10).
Dans cette dernière, en effet, il existe 2 vocabulaires: V_N (variables) et V_T (terminaux) tels que $V_N \cap V_T$ est un ensemble vide. Une généralisation analogue à celle d'un système cladistique peut être trouvée dans les "L Systems" (cf. par exemple Rozenberg ,1974,p.3) : ces systèmes n'utilisent qu'un vocabulaire unique Σ.

Les L-systems constituent un domaine très important dans la théorie des langages formels. Leur but initial (cf. Lindenmayer, 1968 a et b) fut de trouver des modèles mathématiques de croissance d'organismes multicellulairs filamenteux prenant en compte l'interaction entre les cellules et des changements possibles dans l'environnement. Dans la suite, ils furent utilisés par exemple pour modéliser la croissance et la floraison de plantes supérieures telle l'aster (cf. Frijters et Lindemayer, 1974).
Dans un système cladistique, la distinction d'un vocabulaire terminal et d'un vocabulaire non-terminal est non pertinente. En effet, le concept de FIN de l'évolution n'existe pas dans le modèle cladistique. Dans ce dernier, tous les OTUs qu'ils soient actuels ou fossiles (avec ou sans descendance) sont d'ailleurs mis sur le même pied. Et les OTUs à analyser sont considérés comme terminaux. Les terminaux varient donc en fonction du problème à résoudre.
D'autre part, il faut noter qu'en analyse cladistique, le vocabulaire pourra ,en toute généralité, aussi bien être nominal (cf. le problème de Steiner, par exemple) que numérique.

En ce qui concerne P,les remarques suivantes peuvent être faites :

1.La définition donnée ci-dessus est adaptée de Rozenberg (1969,p.3).Elle s'applique à un cas particulier de L System puisqu'il suppose l'absence d'interactions et l'existence d'une seule table. Autrement dit, elle est celle qui intervient dans un "0L system", selon la définition de Rozenberg (1969,p.3). Dans un tel système, une production s'écrit selon les notations de Rozenberg (1969 p.5) $a \rightarrow \gamma$ où a est le membre de gauche et γ, le membre de droite (au lieu de $\langle \wedge, a, \wedge, \gamma \rangle$ ou de $\langle \wedge, a, \wedge \rightarrow \gamma \rangle$) .

2.La définition ci-dessus pourrait encore être précisée davantage dans le cadre de l'analyse cladistique car ici $|a| = |\gamma|$ c.-à-d. le nombre de symboles qui composent le mot a (ou longueur de a) est identique au nombre de symboles qui composent le mot γ . Ceci constitue certainement une propriété remarquable de tous les systèmes cladistiques et apparente ces derniers aux grammaires sensibles au contexte (c.-à-d. de type 1, cf. Hopcroft et Ullman, 1968.) .

3. Traduisant quels états de caractère peuvent être obtenus à partir d'un état donné, les règles de réécriture d'un SC incorporent des contraintes relatives à la classification des caractères eux-mêmes (binaires / sans contrainte ; qualitatifs / cladistiques / ordonnés non-dirigés) (cf.tableau 2.1.) mais aussi au type d'approche numérique choisie pour ce caractère (cf. tableau 3.1.). C'est ainsi qu'elles doivent prendre en charge le cas échéant l'interdiction des réversions au sens cladistique (cf. paragraphe 3.1.1.) en tenant compte de l'état à la racine

(cf. type Camin-Sokal dans le tableau 3.2.). De la même façon, elles doivent être en mesure de prohiber toute convergence au sens cladistique (cf.paragraphe 3.1.1.) (par exemple en utilisant un compteur).
De surcroît, il faut qu'elles dirigent la topologie de l'arbre: il leur revient de dicter le degré des sommets.Dans l'exemple donné ici, elles permettront d'obtenir un arbre ternaire.

TABLEAU 3.2 : Table P fournissant la règle de réécriture concernant l'exemple proposé dans le texte.

pour chaque caractère :									
si p est la racine cladistique	o	n	n	n	n	n	n	n	n
si état au noeud p	-	1	0	?	1	0	?	-	-
si type évolutif	-	CS	CS	CS	CS	CS	CS	CS	W
si état primitif	-	0	0	0	1	1	1	?	-
alors									
état au noeud fils de p	?	X	X	X	X	X	X	X	X
soit	X	/	/	/	/	/	/	/	/
soit état au fils gauche de p	X	1	?	?	?	0	?	?	?
et état au fils droit de p	X	1	?	?	?	0	?	?	?

NB. ? = peut être soit 0 soit 1
W = Wagner (sensu stricto)
CS= Camin Sokal
- = quelle que soit la valeur
/ = pas de réécriture.
X = ne s'applique pas.
n = non
o = oui

4. Les productions ne contiennent pas elles-mêmes de processus d'arrêt. Une limite au nombre de réécritures doit être arbitrairement donnée par ailleurs (voir à ce propos la définition de R dans le tableau 3.4). Ceci est vrai aussi bien dans les systèmes cladistiques (pour limiter le nombre de sommets dans l'arbre) que dans les L systems (par exemple pour arrêter le processus de croissance de la plante dans un modèle).

5. Une structure algébrique de semi-groupe se caractérise par le fait qu'elle est pourvue des opérations d'identité et d'associativité.

Enfin, il faut encore souligner que l'axiome ω intervient également dans la définition d'un "L system" mais pas dans celle d'une grammaire formelle standard.

A partir des commentaires ci-dessus, le tableau synthétique suivant peut être dressé :

TABLEAU 3.3 Comparaison des grammaires formelles standards(GFS), d'un système cladistique (SC) et d'un OL system

GFS	SC		OL system
V_N (variables) V_T (terminaux)	V_c	=	Σ
P (production)	P(table)	=	P(table)
S (symbole de départ)	/	=	/
/	ω	=	ω

NB. pour les explications : voir le texte.

b) Soient ,SC un système cladistique
,$x = a_* .. a_1 \in V_c^*$ avec $a_1 , ..., a_n \in V_c$
,$y \in V_c$.

On dira qu'x <u>dérive directement</u> en y dans SC si on a $y = \gamma_1 .. \gamma_n$ avec $\gamma_1 , ..., \gamma_n$ dans V_c^* tel que, pour chaque i dans $\langle 1,..,n \rangle$, P de SC contient une production de la forme $a_i \rightarrow \gamma_i$

4. Les productions ne contiennent pas elles-mêmes de processus d'arrêt. Une limite au nombre de réécritures doit être arbitrairement donnée par ailleurs (voir à ce propos la définition de R dans le tableau 3.4). Ceci est vrai aussi bien dans les systèmes cladistiques (pour limiter le nombre de sommets dans l'arbre) que dans les L systems (par exemple pour arrêter le processus de croissance de la plante dans un modèle).

5. Une structure algébrique de semi-groupe se caractérise par le fait qu'elle est pourvue des opérations d'identité et d'associativité.

Enfin, il faut encore souligner que l'axiome ω intervient également dans la définition d'un "L system" mais pas dans celle d'une grammaire formelle standard.

A partir des commentaires ci-dessus, le tableau synthétique suivant peut être dressé :

TABLEAU 3.3 Comparaison des grammaires formelles standards(GFS), d'un système cladistique (SC) et d'un OL system

GFS	SC		OL system
V_N (variables) V_T (terminaux)	V_c	=	Σ
P (production)	P(table)	=	P(table)
S (symbole de départ)	/	=	/
/	ω	=	ω

NB. pour les explications : voir le texte.

b) Soient .SC un système cladistique
.$x = a_1 .. a_n \in V_c^*$ avec $a_1, ..., a_n \in V_c$
.$y \in V_c^*$.

On dira qu'x <u>dérive directement</u> en y dans SC si on a $y = \gamma_1 .. \gamma_n$ avec $\gamma_1, ..., \gamma_n$ dans V_c^* tel que, pour chaque i dans $\langle 1,..,n \rangle$, P de SC contient une production de la forme $a_i \rightarrow \gamma_i$

Un **langage** généré par un système cladistique L(SC) est par définition l'ensemble de tous les mots (axiome inclus) qui peuvent être dérivés à partir de l'axiome ω dans SC.

NB.1. Ces définitions sont tout à fait analogues à celles données par Rozenberg (1974 p.4 et 5).

2. Il faut bien remarquer le fait que dans les systèmes cladistiques, les réécritures sont absolument parallèles (c.-à-d. que toute occurrence de lettre dans un mot est réécrite en une seule étape de dérivation). Comme le remarque Rozenberg (1974 , p.6), ceci est une propriété des langages générés par des OL Systems alors que les langages générés par les grammaires de type 0 ou "context free" procèdent de réécritures absolument séquentielles, (c-à-d.qu'une seule occurrence d'un seul symbole est réécrite par étape de dérivation).

c) Etant donné un système cladistique SC tel que défini ci-dessus, et n mots de longueur $|\omega|$ utilisant le vocabulaire V_c et décrivant chacun un OTU, reconnaître si les n mots donnés appartiennent au langage L(SC) **(c-à-d) réaliser un "parser") équivaut à construire un arbre phylogénétique enraciné** ou cladogramme.
De plus, étant donné n+1 OTUs, un arbre phylogénétique non enraciné pourrait être décrit comme un arbre pouvant être obtenu en choisissant successivement chacun des n+1 OTUs comme axiome du SC.

Il faut noter qu'un système cladistique est utilisé en pratique pour la reconnaissance de mots (ou parsing). A l'inverse, les L-systems sont généralement employés pour générer des mots. Dans les grammaires formelles standards, aucun des deux sens d'utilisation des règles de réécriture n'est privilégié : reconnaissance et génération de mots sont également considérés.

3.2.2.1.3. Commentaires.

Ainsi donc, un système cladistique tel que défini ci-dessus présente de grandes analogies avec les OL-systems. Cependant, les règles de réécritures sont utilisées dans des sens opposés, selon ces deux systèmes.
D'autre part, un système cladistique a aussi quelques points communs avec les grammaires dépendant du contexte.

Cette approche nouvelle de l'analyse cladistique par le biais des langages formels pourrait peut-être être exploitée par l'utilisation d'automates pour la construction d'arbres phylogénétiques.

3.2.2.2. Définition formelle du problème d'optimisation intervenant dans l'approche numérique de la systématique phylogénétique.

Contrairement à une tendance couramment répandue, il convient de bien faire la distinction entre différentes solutions possibles d'un problème et le problème lui-même.
En effet, avant de proposer une solution sous forme d'algorithme informatisable, il faut avoir clairement posé et explicité le problème : il est donc très utile de dégager la structure abstraite de celui-ci.
Ceci est d'autant plus vrai pour l'inférence d'arbres phylogénétiques qu'il s'agit d'un problème NP-complet (cf. paragraphe 3.2.1.2.2.) et que différentes solutions exactes et heuristiques (cf.paragraphe 3.2.1.2.3.) peuvent lui être apportées.
L'analyse cladistique doit, parmi une multitude d'arbres possibles, sélectionner le(s) meilleur(s). Ceci constitue donc un problème d'optimisation c'est-à-dire un cas particulier de problèmes.
C'est à la définition formelle de ce dernier que le paragraphe ci-dessous va s'attacher.

Une seule formulation mathématique au problème d'optimisation relatif à l'inférence d'arbres phylogénétiques en analyse cladistique (cf. tableau 3.1) va être donnée. Cette définition sera suffisamment générale pour pouvoir être instanciée au problème de Steiner, aux arbres de Wagner (enracinés ou non) et aux méthodes de compatibilité.

La première partie de cette définition utilisera le vocabulaire des langages formels tel que défini ci-dessus et la méthode d'analyse préconisée par Veloso (1982) à la suite de Polya (1957), c.-à-d. "the formulation of problem as a mathematical structure, solutions being certain functions". Ainsi donc, on commencera par définir formellement le domaine des données, D et celui des résultats, R. Ensuite, on formulera une relation binaire, q de D vers R, appelée la condition du problème. Veloso (1982, p.4) explique que, cela étant, une solution du problème est une fonction de D vers R assignant à toute donnée d de D un résultat possible a(d) $\in$ R satisfaisant la condition du problème, c.-à-d. a : D -) R tel que pour tout d $\in$ D, on a (d, a(d)) $\in$ q .
Lorsque D, R et q sont définis, le problème de l'obtention de **tous** les arbres phylogénétiques pertinents, c-à-d respectant D, R et q possible et mathématiquement formulé. Mais ceci est insuffisant puisqu'il faut encore choisir le(s) meilleur(s) de ces arbres.

La seconde partie de la définition particularisera le problème pour en faire un problème d'optimisation.
Il faudra donc ajouter à la structure générale du problème (c-à-d. ⟨R, D, q⟩) une fonction f (critère d'optimisation) et un ensemble L dont les valeurs permettent d'ordonner (dans une certaine mesure) les solutions.
Ainsi donc L doit être muni d'une relation d'ordre. Dans le cas le plus général, lorsqu'il est difficile de comparer les valeurs

Tableau 3.4. L'inférence d'arbres phylogénétiques en analyse cladistique : Définition formelle du problème d'optimisation intervenant dans l'approche numérique.

P.O. = ⟨D, R, q, L, f⟩.

avec

D : Ensemble de tous les ensembles d tels que :
. # de d = N>0
. 1 élément de d est un mot , utilisant des symbôles de V_c d'un SC
. identifiant un OTU

R : Ensemble de tous les arbres :
- comportant 2N-2 (ou 2N-1 ou ...) sommets
- et dont les feuilles sont les N éléments d'un ensemble d de D

q : est telle que l'on puisse pour $d^{**} \in D$

construire r ∈ R

à l'aide des règles de réécritures de F du SC.

L : ensemble des réels >0 (treillis particulier).

f : associe - à chaque élément de r
- un élément de L.

des solutions (par exemple si celles-ci sont des couleurs), il suffira que cette relation d'ordre puisse définir un treillis. (Une définition de cette structure algébrique peut être trouvée dans Goguen et al., 1973, p.36). Dans des cas particuliers, très fréquents, la relation d'ordre sera totale.
Enfin en toute généralité la fonction f associe à chaque r élément de R un élément de L.

La définition générale donnée au tableau 3.4 peut s'appliquer à différents cas de figures.
Dans une analyse de compatibilité, les règles de réécriture interdiront les réversions et la convergence des caractères et la valeur de f sera le nombre de caractères ainsi compatibles, à maximiser.
En revanche, pour les méthodes de parcimonie, la valeur de f sera la longueur de l'arbre à minimiser. Il faut noter que la valeur de (-f) sera courramment (cf. seconde partie de ce mémoire) appelée "parcimonie" de l'arbre.
Dans le cas des arbres de Wagner, les symbôles de V_c seront pris dans le domaine des réels et f sera mesuré dans la métrique de Manhattan définie ci-dessus (voir paragraphe 3.2.1.1.1.2.)

CHAPITRE 4 . INTERET DES CLADOGRAMMES.

"Our classifications will come to be,
as far as they can be so made,
genealogies."
Darwin , 1859 : 486.
(cité dans Hull , 1979.)

Un arbre phylogénétique enraciné , résultat de l' analyse cladistique , présente un intéret fondamental : il contribue simultanément à l'étude de l'évolution et à la systématique.

4.1. ETUDE DE L' EVOLUTION

4.1.1. **Définition préliminaire.**

Spéciation: "processus de formation ou de multiplication des espèces par isolement." (Chaline, 1983a : 30)

4.1.2. **Contribution à l'étude de l'évolution.**

"... the purpose of methods of phylogenetic inference is precisely to study evolution." (Farris, 1986 b :26). Cette affirmation se justifie à différents titres.

L' explication la plus évidente a déja été fournie à la fin du paragraphe 1.2. Un cladogramme traduit des relations généalogiques entre taxons: il met en évidence des groupes-frères, partageant en exclusivité un anc@tre commun hypothétique et il s'exprime "en termes de degré d'ascendance commune" (Tassy, 1986 - texte original non-souligné).

Mais qu'en est-il des relations "anc@tre-descendant" entre OTUs ? Ce sujet est à la fois relativement marginal dans l'école cladiste et complexe. Selon Engelmann et Wiley (1977), ces relations sont des hypothèses non-testables dans un système cladistique car pour pouvoir les réfuter , il faudrait pouvoir se baser sur le partage d'états primitifs de caractères. Si le statut d'anc@tre direct ne peut pas @tre directement démontré, il n'en est pas moins vrai qu'un OTU-anc@tre apparaitra dans un cladogramme comme un élément d'une polytomie ou comme un élément sans apomorphie d'une dichotomie (de Queiroz & Donoghue, 1988 : 332). Dès lors , à partir d'un cladogramme, il est possible de déduire un ensemble d'arbres (Platnick, 1985 :91) où des relations "anc@tre -descendant" sont exprimées. Ces derniers arbres sont nommés "scénarios" par Nelson (1976). C'est cette

terminologie qui sera retenue ici. Elle n'est cependant pas universellement reconnue. [Platnick (1985) par exemple fait la distinction entre cladogramme et arbre et il réserve le terme d'arbres phylogénétiques pour désigner des scénarios.]
Ce sont surtout les paléontologistes qui sont intéressés par les scénarios. Dans un but explicatif, ils pourront comme le suggère Delson (1976 :437) y incorporer des données supplémentaires comme le temps, l'écologie ,la distribution géographique etc...
Un complément de bibliograpie concernant les scénarios et leurs relations avec les cladogrammes est donné par Platnick (1985 :91).

Après avoir abordé le sujet de la nature des relations de parenté dans les résultats de l'analyse cladistique, on peut s'interroger sur les rapports existant entre cette analyse et les <u>théories de l'évolution</u> .La réponse à cette question importante présente plusieurs facettes , dont certaines font l'objet de controverses. Le point de vue adopté ici sera celui de Farris (1986 b) car c'est celui qui m'a paru le plus critique et le plus pertinent. Il ne concerne que la méthode de parcimonie : Farris ne reconnait pas comme justifiée la technique de compatibilité (cf. paragraphe 3.1.3) . Cet auteur souligne que l'analyse phylogénétique, contrairement aux méthodes des phénéticiens (cf. paragraphe 4.2.2), ne postule pas la constance de la vitesse de l'évolution. Une illustration de cette différence entre les deux techniques peut être trouvée chez Felsenstein (1983 :325) et chez Farris (1971 : 283-286).Cette distinction méthodologique n'est ,à ma connaissance, contestée par aucun cladiste, bien au contraire ! Et Farris (1985a : 134) en conclut :"When a model of rate constancy is used to justify phenetic clustering as an evolutionary method , and when that method yields an unparsimonious arrangement , consequently, the data call the model itself into question."

En revanche ,Farris (1986 b : 20 & 21) s'oppose notamment à Sæther (1986) qui s'appuie lui-même sur Felsenstein (1978 & 1983),lorsqu'il montre que c'est un non-sens d'affirmer que le critère de parcimonie suppose la rareté de l'homoplasie. Cette prise de position de Farris peut être rapprochée de la distinction à faire (cf. Kluge dans Farris,1985 : 296 et Thompson,1986 :46) entre la parcimonie en tant que hypothèse au sujet de la nature et la parcimonie comme principe méthodologique. Elle s'inscrit également dans la ligne d'autres affirmations de Farris : "The supposed dependence of parsimony analysis on parsimony of evolution has been among the most popular themes of antiphylogenetic writers."(Farris , 1989 : 107)
ou : "...most parcimonious cladograms are those which 'best fit' data at hand. "(cité dans Mickevich & Platnick [1989 :45] à propos de Farris ,1983)
ou "... the wheight of evidence principle by itself entailed the parsimony criterion" (Farris et Kluge , 1986 : 303)
ou encore : "Any alternative phylogeny would be weaker as a theory precisely because it would conform less well with data."(Farris ,1985 : 134.)

De manière plus générale, Farris ne rattache l'analyse cladistique à aucun modèle évolutif particulier (cf. 1985a : 133). Selon lui (1986 b : 26), cette analyse fait les faibles hypothèses suivantes : l'évolution est une succession de générations avec des modifications ("descent with modification") et des similarités observées entre taxons sont héritées d'un ancêtre commun. Et il ajoute :"But it requires few other suppositions and so it supplies a means of studying phylogeny whitout presupposing the details of the process. It supplies as well a means of testing more detailled process theories" (Le texte original n'est pas souligné.)
Les conclusions de Taylor (1987) vont dans le même sens : "Phylogenetic analysis is essential for historical biology..."
De la même façon, Platnick (1986 : 84) explique que le fait d'exclure de la justification du cladisme toute théorie particulière de l'évolution " a pour but précisement de pouvoir tester un spectre complet de modèles évolutifs."
Cette conviction est partagée par d'autres cladistes (cf. références données par Platnick, 1986 : 84) et s'inscrit dans la tendance parfois qualifiée de cladisme structural ou "pattern cladism" (voir aussi à ce propos le paragraphe 2.2.1.2).

Elle est le résultat d'une évolution par rapport à la pensée originale de Hennig (1966 :19 & 21) : celui -ci se référait en effet à un modèle de spéciation qualifié de dichotomique par Platnick (1986).Ce modèle est encore actuellement considéré comme une hypothèse sous-jacente de l'analyse cladistique par Turner et Chamberlain (1989 : 119). Différentes autres théories ont été invoquées depuis lors par certains cladistes . Ainsi ,par exemple Sæther (1986 : 1) affirme : "The speciation mode termed 'punctuated equilibrium' by Eldredge and Gould (1972) is an implied consequence of the deviation rule of Hennig (1966 :56-65)..."
Quant à Wiley & Brooks (1982 et Brooks et Wiley 1986), ils assimilent 'évolution' et 'entropie', considèrent les espèces comme des 'structures dissipatives' (Brooks et Wiley, 1986 :40) et prétendent que leur théorie est à la base de l'analyse de parcimonie (cf. Farris ,1989 : 106). Le jugement de Farris (1989) à leur sujet est sans appel : il qualifie la théorie de ces auteurs de "utterly baseless analogy" (:108) et déclare :"...Brooks and Wiley seem to have had no other purpose to create the ... false impression that their theory is indispensable to phylogenetic systematics."

Une très belle illustration de l'utilisation de l'analyse cladistique pour tester des modèles évolutifs vient d'être apportée par Mindell et al. (1989). La méthode de ces auteurs consiste à "comparer le nombre de changements évolutifs encourus
1) par des espèces-soeurs , par rapport à un outgroup
2) par des clades-soeurs , par rapport à un outgroup." (Mindell et al. 1989 :50). (Cette mise en parallèle suppose des vitesses d'extinction similaires chez les différents taxons.) Si la spéciation accélère la divergence, alors un plus grand nombre de pas évolutifs doient être observés chez des taxons ayant subi

davantage de divisions de lignées (spéciations). Ainsi par exemple, dans la fig. 4.1 , si l'hypothèse formulée ci-dessus est vraie ," ... la comparaison de l'espèce 1 et de l'espèce 7 révèlera une quantité plus importante de changements par rapport à l'outgroup dans la première espèce...De la même façon, la confrontation des clades-soeurs ayant comme ancêtre commun le noeud A ... montrera davantage de pas dans le clade comportant 5 espèces (1-5) que dans celui de 2 espèces (6-7)..."(Mindell et al. 1989 : 51).

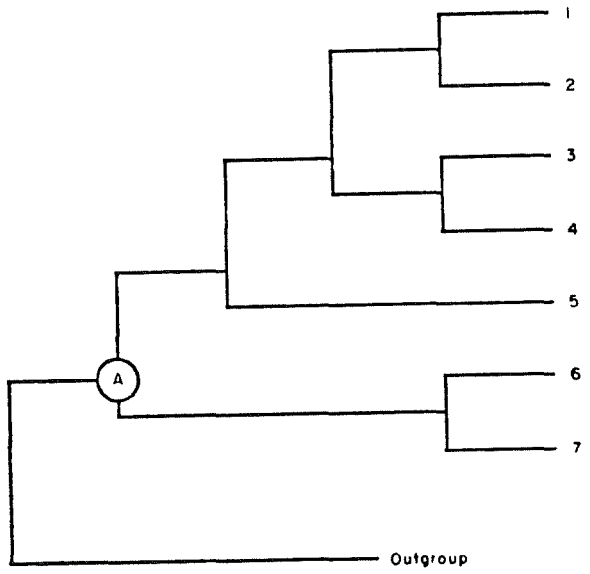


Fig.4.1. "Diagramme schématique illustrant un test phylogénétique de l'évolution associée à la spéciation" (cf. "speciational evolution", selon Mayr ,1988)
(extraite de Mindell et al. 1989 :51)

Mindell;. et al. testèrent "l'hypothèse nulle que les changements ...sont uniformément répartis entre les taxons-frères..." et ils calculèrent "la probabilité d'obtenir la distribution observée par chance , étant donnée cette hypothèse nulle, en utilisant un test binomial."(Mindell et al. 1989 : 52)
A partir d'arbres de Wagner (sensu stricto) orientés, basés sur des allozymes (c.-à-d. des alléloprotéines) et relatifs à Sceloporus (Reptilia), ces auteurs montrèrent ainsi que l'hypothèse du "rôle de la spéciation dans la facilitation des changements évolutifs" (cité dans Mindell E 1989 : 591 à propos de Mayr, 1988) ne peut être rejetée. Ces résultats sont conformes à l'idée centrale formulée dans la théorie des équilibres ponctués (Eldredge & Gould, 1972) à savoir : "significant evolutionary changes arises in coincidence with events of branching speciation, and not primarily through the in toto

transformation of lineages" (Gould, 1982).Les conclusions de Mindell et al. (1989) permettent de rejeter, pour les données analysées, le modèle évolutif plus traditionnel, appelé gradualisme phylétique et ,selon lequel "les changements évolutifs s'accumulent plus graduellement au cours du temps , sans influence marquée des spéciations."(Mindell et al. 1989 : 49.)

4.2. CLASSIFICATION.

4.2.1. **Définitions préliminaires.**

Classification : Très curieusement, le mot classification semble ambigu dans la littérature.
La signification la plus fréquente de ce terme est réservée à ce que Hennig (1966: 194) appelle "a written fixation ...in the form of a catalog or monograph" et correspond à une "hiérarchie Linéenne dans laquelle l'inclusion dans un groupe est indiquée par les rangs et les indentations"(Mickevich et Platnick, 1989:33) [cf. fig. 4.2 (2) et (3)].
Mais on rencontre aussi une sémantique élargie pour ce mot. C'est ainsi que Janvier et al.(1980) en donnent la définition suivante : "En zoologie, action d'ordonner les animaux en groupes c'est à dire de les distribuer par catégories en fonction de leurs relations de parenté."
Cette formulation n'est pas très éloignée de celle de Mickevich et Platnick (1989 : 33): "Classifications convey information about organisms only by specifying a set of terminal taxa and placing them into groups".
Selon la dernière acception , aussi bien une hiérarchie Linéenne qu'un cladogramme sont des classifications. En effet , comme l'expliquent Mickevich et Platnick (1989 : 33): dans un cladogramme, " group membership is indicated by nodes uniting subsets of taxa at particular levels."
Pour éviter toute confusion des deux significations définies ci-dessus , on parlera respectivement , dans ce mémoire, de classification "au sens large" et "au sens strict".

Systématique: " science des classifications des êtres vivants" (Janvier et al., 1980)

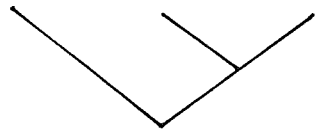
Taxinomie : "...science du système des vivants : disposant les individus en une hiérarchie de classes , elle cherche à décrire l'équilibre des espèces , leurs ressemblances et aussi leurs parentés."
(Benzécri et coll. 1984, 63)
N.B.Selon le dictionnaire Robert, systématique et taxinomie peuvent être tenus pour synonymes.

Embranchement,classe,ordre,famille,tribu,genre,espèce...:
"catégories en usage"
(Benzécri et coll. 1984 : 67.)

Nomenclature zoologique : "application de differents noms à chacun des groupes reconnus de la classification zoologique."
(Janvier et al. 1980)

Subordination et Séquencement: " deux approches cladistes de production formelle d'une classification"(Delson 1977 : 443) (entendez "au sens strict") .
Comme l'explique Delson (1977:443),dans la première approche, les taxons de rang supérieur sont subdivisés en deux ou plusieurs fils de même rang. Dans la seconde: chaque taxon est le frère de tous les taxons de même rang qui le suivent dans la classification. L'illustration suivante de ces deux principes est adaptée de Delson (1977 : 443)

(1)



(2) Superfamille I
Famille A
Superfamille II
Famille B
Famille C

(3) Superfamille I
Famille A
Famille B
Famille C

fig.4.2. : Cladogramme (1) où A ,B et C sont des familles, et classifications (au sens strict) correspondantes basées sur la subordination (2) et sur le séquencement (3).

4.2.2. **Contribution du cladogramme à la systématique.**

Comme signalé ci-dessus, un arbre phylogénétique enraciné peut déjà être considéré comme une classification ,au sens large.

De plus, une classification ,sensu stricto, peut se déduire d'un cladogramme. "Cladistic systematics ... desire that a classification essentialy be a direct transformation of a cladogram into a formal hierarchy." (Delson, 1976 : 442)
Hennig (1966 :237) affirme :" a properly drawn phylogenetic tree must be directly translatable into the languages of phylogenetic systematics." En effet, selon cet auteur, un arbre

phylogénétique (enraciné) se traduit directement en une classification hiérarchique (au sens strict) fondée " sur les groupes monophylétiques subordonnés" (cf. Tassy ,1986 :91). Ainsi, par exemple, dans la figure 4.3 ,les taxons B et C, partageant des états dérivés de caractères, seront nommés le groupe Y dans la classification cladistique (au sens strict).

Cette approche s'oppose radicalement à celle des évolutionistes classiques (cf. Simpson,1961 et Mayr ,1969). Selon cette dernière , ce sont les taxons A et B de la figure 4.3 qui doivent être rassemblés pour former le groupe ancestral ou "souche" W dans une classification au sens strict. "Dans cette exemple, phylogénies cladistiques et évolutionistes sont les mêmes..."(Tassy, 1986 : 89). Mais en réalité, les évolutionistes groupent sur base de partage d'états primitifs de caractères , invoquant le fait qu'ils sont l'"expression d'affinités profondes" (Tassy , 1986 : 88). Ils aboutissent donc à "un schéma phylogénétique inverse du schéma hennigien" (Tassy 1986 :88).

La méthode de classification cladistique (au sens large) est aussi en complet désaccord avec celle de l'école des phénéticiens ou de taxinomie numérique (cf. Sokal et Sneath ,1963). Selon celle-ci, une classification doit être basée sur toutes les similitudes observables ("overal similarities" cf. Farris,1979 :484).

Il apparaît donc que les deux principes fondamentaux qui distinguent l'approche phylogénétique des autres approches sont la monophylie des taxons et le groupement sur base de synapomorphies (cf. Farris,1986 : 15).

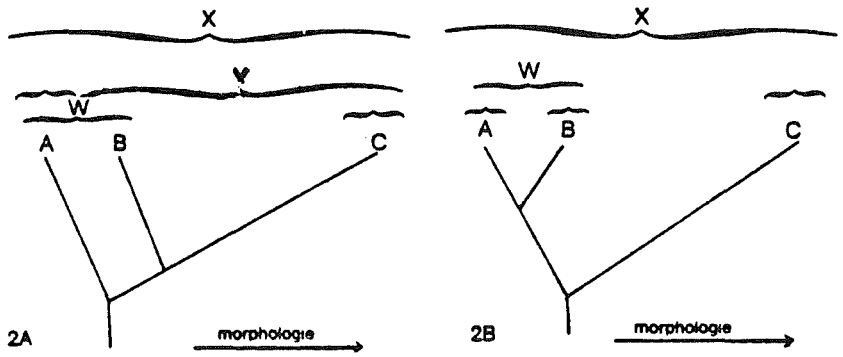


Fig 4.3. (extraite de Tassy 1986 : 90)
"Morphologie , filiation et classification.
I : relations de parenté et divergence morphologique:
le taxon C est éloigné de B en raison d'une forte divergence.
Les taxons B et C forment le groupe Y dans une classification cladistique.
Les taxons A et B forment le groupe W dans une classification morphologique et évolutioniste.
II :dendrogramme phénétique ;les taxons A et B sont étroitement apparentés sur base de la similitude générale et forment un taxon W classifié."

Ainsi donc la méthode de classification préconisée par Hennig se distingue clairement de celle des autres grandes écoles de sytématique. (Un complément d'information à ce sujet peut être trouvé chez Tassy ,1986.) Le cladisme est d'ailleurs apparu comme révolutionnaire, voire même scandaleux !(Thuillier ,1981)

Toujours selon Hennig (1966),la classification (au sens strict) pouvant être déduite d'un arbre phylogénétique est unique et doit être le système de référence universel : il écrit en effet (: 239) ... the phylogenetic system may be regarded, for inherent reasons, as the general reference system of biology."

Récemment ,Tassy (1988) a bien montré que ceci était une utopie : il y a , explique-il (:52), beaucoup de classifications cladistiques (sensu stricto) possibles sur la base d'un seul et unique cladogramme parce que, pour subdiviser un cladogramme, il faut effectuer des choix logiques et décider de la nomenclature.(En particulier, Tassy utilise à la fois la subordination et le séquencement comme base de sa classification des Proboscidiens.) L'auteur (:52) met en évidence le fait que "la méthode de classification formelle en sytématique phylogénétique doit rester pluraliste ". En revanche, il souligne le caractère universel du cladogramme,qui contient toutes les informations et qui peut servir de base de discussion unique aux systématiciens. Il affirme en effet (1988 : 52): "...this challenge has led to the cladogram as the most important contribution in the field of systematics... In other words, Hennig's 'phylogenetic system' should be synonymized with 'cladogram',not with 'classification'."

4.3.JUSTIFICATION DE L'ANALYSE CLADISTIQUE.

4.3.1. **Définitions préliminaires.**

Diagnose: détermination des caractéristiques d'un taxon.
(Définition adaptée de Wiley, 1981 : 377.)

Cladogenèse :multiplication des lignées par séparation.
(Définition adaptée de Chaline ,1981 : 47.)

Anagenèse : "progression évolutive ou évolution phylétique".
(Chaline,1981 : 47.)
ou encore "changes in morphology along a lineage"
(Stebbins & Ayala, 1985 : 61).

4.3.2. **Capacité de description et pouvoir explicatif**

Pour terminer cette première partie du mémoire comprenant un revue de la littérature cladistique,il est opportun de synthétiser les justifications de l'analyse phylogénétique.Il sera fait référence essentiellement aux articles de Farris (1979,

1980a , 1980b , 1982b , 1983, 1985a, 1989) car ce dernier est un des défenseurs les plus ardents et les plus convainquants du cladisme et en particulier de l'analyse de parcimonie (cf. paragraphe 3.1.3 .
Sober (1985: 209) souligne avec raison que Farris ne situe pas son plaidoyer en faveur de la parcimonie à un niveau statistique (contairement à lui-même qui justifie cette méthode sur base du maximum de vraisemblance).
Quant à Platnick (1986 :83), il fait l'éloge et le résumé de l'argumentation de Farris en ces termes : "... unexcelled justifications of cladistics , namely the equivalence between classifications with maximal descriptive ability and those with maximal explanatory power."

Ceci mérite d'être détaillé ! Farris (1982b : 416) fournit lui-même les grandes lignes d'une synthèse de sa pensée. Il commence par y rappeler qu'en 1979, il a démontré formellement qu'" une hiérarchie est capable de fournir une descrition efficiente des caractères dans la mesure où chacun des états de ces caractères n'intervient dans la diagnose que de peu de taxons"(Farris 1982b : 416). S'appuyant sur la théorie de l'information (1979 : 506),il a souligné que "des classifications efficientes en ce sens facilitent la description et le stockage de l'information relative aux caractères vu qu'elles minimisent la redondance intervenant dans la distribution des états des caractères : les glandes mammaires doivent être associées uniquement aux mammifères et ne doivent pas être reprises séparément pour chacun des mammifères"(Farris 1982b :416).
Pour la même raison ,des hiérarchies efficientes permettent aussi "d'incorporer l'information d'une matrice de caractères avec un nombre minimal de noms de groupe de telle sorte que le contenu informatif au sujet des caractères est maximal par groupe"(Farris 1982b : 416)."Et Farris (1979 : 483) explique : " Classifications the description of data in the fewest symbols . Most parsimonious trees have highest information content,..., allow the data to be completely summarized by the most succint diagnoses , minimize exceptions to such complete diagnoses..."
Cet auteur a également démontré que les méthodes phylogénétiques de classification contiennent davantage d'information sur les distances (Farris ,1979) et donc sur les degrés de divergence (Farris 1980 b) entre taxons que celles des évolutionistes et des phénéticiens. Il en conclut qu'elles sont "plus efficaces pour représenter aussi bien l'anagenèse que la cladogenèse" (Farris 1980b :387).
Ainsi donc , selon Farris, la <u>capacité de description</u> est plus grande dans l'approche cladiste que dans les autres classifications,aussi bien en ce qui concerne les caractères que les distances entre taxons.

Ultérieurement, cet auteur (1983) montra aussi que cette approche coïncide avec un <u>pouvoir explicatif</u> maximal. En effet, il souligna qu'une "hypothèse de relation phylogénétique est capable d'expliquer des similarités observées entre organismes dans la mesure où elle les identifie comme étant héritées d'un ancêtre commun." (Farris 1982b :416). Des similarités ne peuvent pas s'expliquer de cette façon dans la mesure où l'arbre présummé

recquiert des hypothèses d'homoplasie. Ainsi donc :" Each additional requirement for a separate origin of a feature reduces the explanatory power of a theory of phylogenetic relationship.(Farris &Kluge , 1986 : 300).
Et dès lors,"l'application du critère de parcimonie , choisissant l'arbre qui minimise le recours aux hypothèses d'homoplasie , fournit donc les schémas dont le pouvoir explicatif est maximal."(Farris 1982 b : 416). D'autre part, Farris (1983 et 1980a) a mis en évidence que "la présence d'un trait donné dans les diagnoses de plusieurs groupes correspond à l'interprétation de l'origine multiple de celui-ci." (Farris 1982b :416).En revanche , lorsqu'une caractéristique partagée par plusieurs taxons n'apparaît que dans la diagnose d'un seul , cela revient à expliquer les similitudes observées comme étant héritée d'un ancêtre commun.
Et Farris (1982b : 416) de conclure : "Il y a donc une correspondance directe entre la capacité descriptive d'une classification et son pouvoir explicatif comme hypothèse phylogénétique."

SECONDE PARTIE	Etude approfondie et comparée d'un algorithme heuristique d'inférences d'arbres phylogénétiques basée sur le principe de parcimonie.

Après avoir passé en revue des éléments essentiels de la littérature cladistique et défini formellement le problème général de l'inférence d'arbres phylogénétiques en systématique phylogénétique, il convient maintenant de se tourner vers les solutions apportées à ce problème, c'est-à-dire les algorithmes* contenus dans les logiciels disponibles. L'analyse qui va suivre se focalisera essentiellement sur l'algorithme de recherche heuristique des topologies les plus parcimonieuses utilisées par le programme MIX (3.1) écrit par Felsenstein pour micro-ordinateurs

CHAPITRE 5 : INTRODUCTION.

5.1. JUSTIFICATION DE CETTE ETUDE

Prenant la relève des programmes pour gros ordinateurs (voir étude comparative de Luckow et Pimentel 1985), les logiciels d'analyse phylogénétique pour micro-ordinateurs prennent une importance croissante et constituent un domaine hautement compétitif. Déjà dans un rapport sur le Congrès de la Société Willi Hennig, Coddington (1987,p.179) remarquait: "Obviously, mainframe Computers are still desirable for large or complex data sets, but mainframe programs will probably lag behind micro versions in terms of innovations." Au dernier Congrès de cette même société qui s'est tenu en 1988, seuls des programmes pour micro-ordinateurs furent présentés et discutés. Et Tassy et Darlu (1988, p.295) affirment: "Nous n'en sommes actuellement qu'au début de l'utilisation de la micro-informatique à des fins d'analyse phylogénétique... Au vu de l'essor actuel de la micro-informatique il y a fort à parier que l'ordinateur personnel deviendra l'outil habituel sinon privilégié du «Phylogénéticien».".C'est donc vers les logiciels d'analyse phylogénétique pour micro-ordinateurs que je me suis tournée.

Parmi ceux-ci, je me suis particulièrement intéressée à ceux employés en paléontologie c'est-à-dire qui traitent des données morphologiques. J'ai préféré ceux qui pratiquent une analyse de parcimonie (car cette méthode est très largement utilisée actuellement et est très éloquemment défendue par Farris - cf. chapitre 4) et en particulier qui construisent des arbres de Wagner (au sens large) (car ces derniers constituent une pierre

---

* algorithme : déformation du nom de "l'algébriste Al-Khwarizmi (IX[e] siècle)" (Dieudonné, 1987 :62).
"Dans une formulation moderne, on peut définir le terme "algorithme" comme toute chose qui peut être exécutée par un ordinateur." (Goldschlager et al. ,1986 :89)

angulaire de l'analyse cladistique). Vu que les problèmes de parcimonie ont été démontrés NP-complets (cf.paragraphe 3.2.1.2), je me suis orientée vers les algorithmes heuristiques.

Mais je me suis alors trouvée confrontée à des problèmes pratiques.

En effet il n'y a dans la littérature que très peu d'informations concernant ces algorithmes alors que ceux-ci deviennent de plus en plus sophistiqués.

La publication de base concernant des algorithmes heuristiques d'inférence d'arbre de Wagner est déjà ancienne (1970) et est due à Farris. Ces algorithmes de 1970 sont très souvent cités et il en existe des explications (complètes ou partielles) du fonctionnement "à la main" sur des exemples (Wiley 1981, p.182-188 et Brooks,1984). Mais selon Swofford (1985,p.1-2), ils ne fonctionnent pas bien et Farris lui-même les a apparemment abandonné dès son programme intitulé Wagner 78.
Et Swofford ajoute "Unfortunately, Farris never bothered to tell this to the users of his programs, many of whom persist in citing the 1970 as the method employed."! On peut quand même trouver sous la plume de Farris (1985,p.294) cette petite note relative à son programme d'optimisation (Farris 1970) et à la méthode utilisée par Kluge et Farris (1969):"...both those techniques are by now rather dated..."
Quant aux stratégies heuristiques appelées "branch swapping" (ou "swapping" ou encore réarrangements) qui sont couramment utilisées dans les programmes récents, elles ne sont actuellement l'objet d'aucunes publications détaillées. Felsenstein en parle de manière "vague" (selon ses propres termes) dans les commentaires de PHYLIP. Les renseignements fournis à ce sujet par Swofford dans PAUP user's manual (1985) sont également très succincts.
A défaut d'être publiées, les informations concernant les heuristiques actuellement employées pourraient être cherchées dans les programmes-sources eux-mêmes.
Malheureusement, les concepteurs de programmes d'analyse cladistique ne paraissent vraiment pas disposés à communiquer leurs codes-sources. Une des raisons pouvant expliquer cette attitude est la suivantes : ces programmes sont, pour la plupart, non pas donnés, mais vendus aux chercheurs.
Felsenstein fait toutefois exception à cette règle générale. Il a eu la grande amabilité de me communiquer la dernière version (3.1) de son vaste programme PHYLIP. Ce dernier contient des notices explicatives très volumineuses sous forme de documentation (MAIN-DOC, MIX-DOC et DISCRETE-DOC notamment) et sous forme de commentaires au sein des programmes eux-mêmes. Celles-ci ne paraissent pas toujours mises à jour. De plus PHYLIP est dépourvu de spécifications informatiques.

Il est donc apparu que la manière la plus intéressante d'entrer dans le vif du sujet, était de choisir la partie essentielle d'un des programmes heuristiques de PHYLIP et d'en faire une analyse détaillée et critique (cf. chapitre 6). Parmi ces programmes de PHYLIP,conformément au conseil de PLATNICK (1987),MIX a été préféré à METRO (utilisant les méthodes appelées "Metropolis simulated annealing").

L'analyse algorithmique sera précédée d'une mise au point de vocabulaire et d'une brève présentation de MIX.

5.2. DEFINITIONS ET VOCABULAIRE.

5.2.1. **Définitions préliminaires.**

Topologie d'un arbre phylogénétique : "structure arborée" (cf Barthélemy et Guénoche 88, p. 169).

NB : 1. La topologie reflète donc "un ensemble déterminé de relations cladistiques" (selon les termes de Farris 1970, p. 90).

2. Pour comparer les topologies, il faut bien garder à l'esprit les 2 faits suivants :
- l'identifiant d'un HTU est arbitrairement choisi
- fils gauche et fils droit peuvent être intervertis.

Une topologie donnée peut donc se présenter graphiquement sous différentes formes.

Optimiser une topologie donnée : étant donné cette topologie, minimiser la longueur de l'arbre en recherchant pour elle l'ensemble optimal d'HTUs. (définition adaptée de Farris 1970, p. 90.)

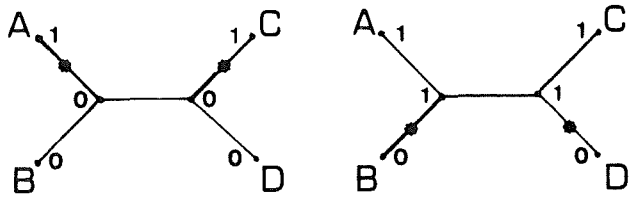


Fig.5.1. Illustration de la possibilité d'existence de plusieurs ensembles optimaux d'HTUs pour une topologie donnée. Etant donné une topologie d'un arbre ternaire, différentes affectations possibles d'état d'un caractère donné aux HTUs (et corrélativement différentes localisations des changements évolutifs symbolisés par les astérisques). [adapté de Felsenstein 1983,p.315]

Remarques 1."La procédure d'optimisation d'une topologie peut être utilisée pour accroître la parcimonie de l'arbre ... (traduit de Farris 1970,p. 92.)

2. Il arrive très fréquemment qu'il y ait plusieurs ensembles optimaux d'HTUs pour une topologie donnée (voir figure 5.1)

5.2.2. **Remarques d'ordre terminologique.**

MIX, le programme est à distinguer de sa documentation, MIX.DOC (non-reprise ici). Dans la description MIX, le vocabulaire utilisé, relatif à l'analyse cladistique, sera celui défini dans la première partie de ce mémoire.
A ce propos, le lecteur est invité à bien noter la signification réservée ici au terme "variable" (cf. paragraphe 2.1).
Il doit aussi savoir que ce vocabulaire ne coïncide pas parfaitement avec celui utilisé par Felsenstein dans la documenta-tion de PHYLIP. Ainsi ce qui a été défini au paragraphe 1.1 comme un OTU est appelé espèce par le concepteur de MIX. D'autre part ce dernier ne fait pas la distinction entre caractère et variable telle qu'elle est décrite au paragraphe 2.1. Enfin, Felsenstein évoquera plus généralement la maximisation de la parcimonie (ou de la vraisemblance) de l'arbre plutôt que la minimisation de sa longueur.

5.3. LE PROGRAMME MIX.

Le programme MIX, dont l'algorithme de recherche des topologies les plus parcimonieuses va être analysé, est rédigé en Pascal ("MIX.PAS"). Il se trouve repris dans l'annexe III de ce mémoire.

MIX a été implémenté sur VAX/VMS aux Facultés Notre-Dame de la Paix, puis compilé et exécuté avec différents jeux de données pour tester les diverses options analysées dans ce travail (cf. paragraphe 5.4.1). Ces exécutions ont également eu pour but de mettre en évidence certaines subtilités non-explicitées par Felsenstein : possibilité de différence entre racine cladistique et racine de construction de l'arbre (cf. paragraphe 6.5.3 et fig. 5.3), traitement du polymorphisme (cf. paragraphe 6.5.1 et fig. IV.1) ... A partir de nombreux "runs" de MIX qui ont été réalisés, quelques "output" intéressants et pas trop longs ont été sélectionnés pour figurer dans ce travail (cf. fig. 5.2, 5.3 et 5.4 ainsi que l'annexe IV).

Selon la terminologie utilisée ici, MIX apparaît comme un programme réalisant l'inférence d'arbres de Wagner (au sens large) à partir de variables de type Wagner (sensu stricto) ou

Remarques 1."La procédure d'optimisation d'une topologie peut être utilisée pour accroître la parcimonie de l'arbre ... (traduit de Farris 1970,p. 92.)

2. Il arrive très fréquemment qu'il y ait plusieurs ensembles optimaux d'HTUs pour une topologie donnée (voir figure 5.1)

5.2.2. **Remarques d'ordre terminologique.**

MIX, le programme est à distinguer de sa documentation, MIX.DOC (non-reprise ici). Dans la description MIX, le vocabulaire utilisé, relatif à l'analyse cladistique, sera celui défini dans la première partie de ce mémoire.
A ce propos, le lecteur est invité à bien noter la signification réservée ici au terme "variable" (cf. paragraphe 2.1).
Il doit aussi savoir que ce vocabulaire ne coïncide pas parfaitement avec celui utilisé par Felsenstein dans la documenta-tion de PHYLIP. Ainsi ce qui a été défini au paragraphe 1.1 comme un OTU est appelé espèce par le concepteur de MIX. D'autre part ce dernier ne fait pas la distinction entre caractère et variable telle qu'elle est décrite au paragraphe 2.1. Enfin, Felsenstein évoquera plus généralement la maximisation de la parcimonie (ou de la vraisemblance) de l'arbre plutôt que la minimisation de sa longueur.

5.3. LE PROGRAMME MIX.

Le programme MIX, dont l'algorithme de recherche des topologies les plus parcimonieuses va être analysé, est rédigé en Pascal ("MIX.PAS"). Il se trouve repris dans l'annexe III de ce mémoire.

MIX a été **implanté** sur VAX/VMS aux Facultés Notre-Dame de la Paix, puis compilé et exécuté avec différents jeux de données pour tester les diverses options analysées dans ce travail (cf. paragraphe 5.4.1). Ces exécutions ont également eu pour but de mettre en évidence certaines subtilités non-explicitées par Felsenstein : possibilité de différence entre racine cladistique et racine de construction de l'arbre (cf. paragraphe 6.5.3 et fig. 5.3), traitement du polymorphisme (cf. paragraphe 6.5.1 et fig. IV.1) ... A partir de nombreux "runs" de MIX qui ont été réalisés, quelques "output" intéressants et pas trop longs ont été sélectionnés pour figurer dans ce travail (cf. fig. 5.2, 5.3 et 5.4 ainsi que l'annexe IV).

Selon la terminologie utilisée ici, MIX apparaît comme un programme réalisant l'inférence d'arbres de Wagner (au sens large) à partir de variables de type Wagner (sensu stricto) ou

de type Camin-Sokal. Par souci de concision, il sera omis de préciser dans la suite que ce premier type doit se comprendre au sens strict. Les variables de Mix peuvent prendre les états suivants: {**0** ,**1** ,**?** (c.-à-d. inconnu), **P** (ou B c.-à-d. polymorphe c.-à-d. 0 et 1)}.Ce dernier état peut prêter à confusion (cf.paragraphes 6.5.1.5 et 6.5.1.6). D'autre part,il n'est pas numérique et par conséquent son intervention dans le calcul de la longueur de l'arbre n'est pas direct. Par contre l'état "?" peut être considéré comme numérique car il pourra être remplacé soit par 0 soit par 1 (c.f.paragraphe 6.5.2.3).Sans comptabiliser la valeur "?" (qui apparaît en fait comme un artifice) ni la valeur P (qui est une singularité de MIX), les caractères eux-mêmes peuvent être qualifiés de binaires (cf.documentation de MIX fournie par Felsentstein) ou encore sans contraintes et ordonnés (cf. paragraphe 2.1).Dans ce dernier cas, ils doivent subir un codage additif binaire (cf. programme FACTOR introduit dans PHYLIP à cet usage) avant d'être introduits comme variables dans MIX et ils ne peuvent prendre la valeur P. Les caractères non-ordonnés ou qualitatifs ne sont pas autorisés.

5.4. LES ENTREES ET LES SORTIES DU PROGRAMME MIX.

5.4.1. **Input de MIX**

* 1ère ligne obligatoire :nombre d'OTUs ("spp") et nombre de variables binaires (chars)(+ symbôles éventuels des options commandées).

* ligne(s) suivante(s) facultative(s): option(s) commandée(s) par l'utilisateur.

Remarque : Les options les plus importantes pour la compréhension du programme MIX sont :

1. l'option J : permet à l'utilisateur d'utiliser un générateur de nombres aléatoires pour choisir l'ordre de prise en compte des OTUs par l'algorithme de construction de l'arbre binaire. (Un "output" obtenu en invoquant cette option est donné dans la fig. 5.2 .)

2. l'option A : permet d'imposer la valeur de l'état primitif (1,0 ou ?) de chaque variable. (Un "output" résultant de la sélection de cette option se trouve repris dans la fig. 5.3.)
(NB : Si la valeur de l'état primitif d'une seule variable est connue au départ, l'arbre sera automatiquement construit enraciné cladistiquement !).

3.l'option O : permet à l'utilisateur d'imposer l'OTU-outgroup. Cette option ne sera prise en compte que si l'arbre binaire construit dans la première étape de l'algorithme est non enraciné cladistiquement (c.-à-d. uniquement si toutes les variables sont de type Wagner et que l'état primitif de chacune d'elles est inconnu). Cette option O présente l'énorme avantage de permettre à l'utilisateur de ne faire aucune hypothèse concernant les états primitifs : c'est le programme qui déduira lui même la solution la plus parcimonieuse compte tenu de la position imposée à la racine cladistique.
(Un exemple de résultat obtenu en ayant choisi l'option O peut être trouvé dans la fig. 5.4).

4.l'option M : permet à l'utilisateur d'imposer le type Camin-Sokal (S ou C) ou Wagner (W ou ?) pour chacun des variables. (cf. l'output de la fig. IV.2 .)

5.l'option S (ou C) permet à l'utilisateur d'imposer le type Camin-Sokal à toutes les variables.

Les options T (threshold), U (user tree), W (weights)et Y ne sont pas fondamentales : elles ne seront pas explicitées dans la présente analyse.

* Matrice de données proprement dites :
(obligatoire)
nom de chaque OTU \ état de chaque variable

Cette matrice est reproduite en début de chaque "output" (cf. fig. 5.2 à 5.4) .

Les renseignements ci-dessus peuvent être retrouvés (sous une forme différente) dans la documentation (MAIN-DOC, MIX-DOC et DISCRETE-DOC) fournie par Felsenstein. Mais il convient d'y ajouter les commentaires synthétiques ci-dessous.

Remarque 1. Les options par défaut du programme MIX sont les suivantes :

1.toutes les variables sont de type Wagner
2.l'état primitif de chacune d'elle est inconnu
} arbre non enraciné cladistiquement !

3.(par convention, le premier fils gauche issu de la

racine binaire est le premier OTU dans l'ordre d'input).

Remarque 2. En résumé, l'état primitif d'une variable peut être, au départ :

0 : soit donnée fournie par l'utilisateur (option A)
soit option par défaut si la variable a été déclarée Camin-Sokal (par option M ou option S).
1 : seulement si imposé comme tel par l'utilisateur (option A)
? : seulement si inconnu (soit déclaré comme tel par l'utilisateur soit conformément à l'option par défaut de MIX).

L'état primitif ne peut être P. De plus, comme le fait remarquer Felsenstein : en utilisant simultanément l'option A et l'option C , l'utilisateur peut imposer un état primitif "?" à un variable de type Camin-Sokal. Mais ce que l'auteur ne dit pas c'est que cela est impossible avec la conjonction des options A et M. (Un listing prouvant cela se trouve repris à la fig. IV.2 : en effet , ce résultat a été obtenu alors que l'état ancestral de la variable de type Camin-Sokal avait reçu la valeur "?" à l'"input". Le programme a forcé cet état à "0".)

Remarque 3. Dès l'input, la procédure inputdata (cf. p.III.7) du programme MIX :

- crée 2* spp-1 (="nonode") sommets de degré 0
- leur attribue un numéro identifiant (qui est le numéro d'ordre d'input pour les OTUs)
- et associe aux spp sommets-OTUs un état (0, 1, ?, ou P) pour chacune des variables.

La manière dont les sommets sont implémentés ne sera pas explicitée ici. En effet, la description qui va suivre veut atteindre un niveau relativement conceptuel.

```
Mixed parsimony algorithm, version 3.1

 5 species,   6  characters

Wagner parsimony method

► Random number seed =      1353

Name          Characters
----          ----------

alp           11011 0
bet           11000 0
gam           10011 0
del           00100 1
eps           00111 0

Adding species:
   bet
   del
   alp
   eps
   gam
Doing global rearrangements
   .........

One most parsimonious tree found:



  alp
  !
  !   eps
  !   !
  !   4--del
  !   !
  !   3-----gam
  !   !
  1--2--------bet

  remember: this is an unrooted tree!

► requires a total of       8.000

steps in each character:
      0   1   2   3   4   5   6   7   8   9
    *-------------------------------------
  0!        1   1   1   2   2   1

From     To      Any Steps?     State at upper node
                                ( . means same as in the node below it on tree)

                 1                  11011 0
1      alp                no       ..... .
1        2                no       ..... .



2        3                yes      .0... .
3        4                yes      0.1.. .
4      eps                no       ..... .
4      del                yes      ...00 1
3      gam                no       ..... .
2      bet                yes      ...00 .
```

Fig. 5.2 "Output" de MIX illustrant le choix de l'option J et l'obtention d'un arbre non-enraciné cladistiquement.

```
Mixed parsimony algorithm, version 3.1

 3 species,   6  characters

Wagner parsimony method

► Ancestral states:
              00000 0


Name          Characters
----          ----------

un            11011 0
deux          11CC0 0
tri           10011 0


Adding species:
   un
   deux
   tri
Doing global rearrangements
   .....

One most parsimonious tree found:



  tri
  !
  2--un
  !
  1-----deux
 /
/


requires a total of       5.000

steps in each character:
          0   1   2   3   4   5   6   7   8   9
      *---------------------------------------
    0 !       1   2   0   1   1   0

From     To      Any Steps?    State at upper node
                               ( . means same as in the node below it on tree)

► root       1          yes     1?... .
  1          2          yes     .?.11 .
  2        tri        maybe     .0... .
  2        un         maybe     .1... .
  1        deux       maybe     .1... .
```

Fig. 5.3 "Output" de MIX illustrant le choix de l'option A, l'obtention d'un arbre enraciné cladistiquement de façon primaire et la mise en évidence de la différence la racine cladistique et la racine de construction.

```
 5 species,   6  characters

wagner parsimony method


Name          Characters
----          ----------

alp           11C11 0
bet           11000 0
► gam         10C11 0
del           00100 1
eps           00111 1


Adding species:
   alp
   bet
   gam
   del
   eps
Doing global rearrangements
   .........

One most parsimonious tree found:



  gam
  !
  !  eps
  !  !
  !  4--del
  !  !
  !  !      alp
  !  !      !
  2--3------1--bet

  remember: (although rooted by outgroup) this is an unrooted tree!


requires a total of       3.000

steps in each character:
         0   1   2   3   4   5   6   7   8   9
     *---------------------------------------
    0!       1   1   1   2   2   1

best guesses of ancestral states:
       0 1 2 3 4 5 6 7 8 9
     *--------------------
    0!   1 0 0 1 1 0

From     To      Any Steps?     State at upper node
                                ( . means same as in the node below it on tree)


root       2          no       ..... .
  2     gam           no       ..... .
  2       3           no       ..... .
  3       4           yes      0.1.. 1
  4     eps           no       ..... .
  4     del           yes      ...00 .
  3       1           yes      .1... .
  1     alp           no       ..... .
  1     bet           yes      ...00 .
```

Fig. 5.4 Illustration du choix de l'option O et de l'obtention d'un arbre enraciné cladistiquement de façon secondaire.

5.4.2. **Output de MIX.**

L'output de MIX comprend une liste d'arbres également parcimonieux. (Voir fig. IV.2 le cas où la liste comprend plus d'un élément .) MIX. DOC souligne que ceux-ci seront imprimés soit enracinés (comprenez "cladistiquement" enracinés) soit non-enracinés (comprenez "cladistiquement" non-enracinés) selon le cas échéant. Si l'on regarde les différents outputs possibles, on s'aperçoit qu'en réalité un arbre non-enraciné cladistiquement est imprimé sous la forme d'un arbre binaire, avec, il est vrai, une sentence "this is an unrooted tree". (cf. fig. 5.2)
Mais un arbre enraciné par "outgroup" se présente exactement sous la m@me forme qu'un arbre non-enraciné cladistiquement. En revanche, la phrase explicative est différente mais elle est ambiguë ! "(although rooted by outgroup) this is an unrooted tree !". (cf. fig. 5.4)
Quant enfin à l'arbre enraciné par anc@tre, il est imprimé de façon légèrement différente du précédent : une ar@te est curieusement rajoutée à l'ancienne racine binaire, augmentant ainsi d'une unité le degré de ce noeud. (cf. fig. 5.3) (En ce qui concerne l'enracinement : des spécifications peuvent être trouvées au paragraphe 6.3.2.1 et une discussion critique au paragraphe 6.5.3.).
D'autre part, l'output de MIX comprend aussi , pour chaque arbre, un tableau donnant le nombre de changements d'états requis pour chaque variable et un autre tableau indiquant pour chaque branche "s'il y a des changements évolutifs qui y sont associés et quels sont les états des variables au niveau de son sommet distal". (Pour une topologie et une parcimonie données, il pourra éventuellement y avoir plusieurs affectations différentes possibles aux états de variables de certains HTUs - cf. fig. 5.1). De plus si l'arbre est enraciné cladistiquement et si au moins un état primitif était inconnu au départ, alors un vecteur donnant l'état primitif le plus parcimoneux pour chaque variable est également imprimé. (cf. par exemple la fig. 5.4)

Il faut bien remarquer (cf. MAIN-DOC) que dans les arbres imprimés par MIX, la longueur d'une ar@te n'est pas proportionnelle au nombre de changements évolutifs qui y sont assignés.
Ceci a pour conséquence, comme le fait remarquer Platnick (1987,p.123), que l'utilisateur devra lui m@me supprimer manuellement les ar@tes de longueurs nulles. Il devra ensuite comparer ces topologies "corrigées" (voir remarque 2 relative à la topologie au paragraphe 5.2.1) et il est possible qu'il constate alors de la redondance dans l'output, plusieurs arbres étant devenus identiques. Ce désagrément potentiel est vivement critiqué par Platnick (1987:143), dans sa comparaison des programmes de parcimonie pour micro-ordinateurs.

CHAPITRE 6.ANALYSE DE L'ALGORITHME DE RECHERCHE HEURISTIQUE DES TOPOLOGIES LES PLUS PARCIMONIEUSES UTILISE PAR LE PROGRAMME MIX DE FELSENSTEIN.

6.1. AVANT-PROPOS.

Pour aborder la description qui va suivre il est impératif d'avoir clairement à l'esprit la distinction déjà signalée entre racine cladistique et racine au sens de la théorie des graphes (qui pourra être nommée ici simplement "racine").

Cette description concerne strictement la recherche des différentes topologies les plus parcimonieuses. Elle ne s'intéresse donc pas aux différentes affectations possibles d'états des variables aux ancêtres hypothétiques pour une topologie donnée, de longueur déterminée. Elle ne prend pas non plus en compte les modalités de mémorisation ni d'impression des différentes topologies sélectionnées. En clair, elle se concentre donc sur l'essentiel de la première partie de la procédure maketree de MIX (cf. page III.8).
D'autre part, ainsi que cela a déjà été précisé à propos de l'input , elle se focalise sur les cas de base. En particulier, elle passe l'option T sous silence. Ceci a pour conséquence que la valeur du critère à optimiser est non pas réelle (cf. implémentation de Felsenstein) mais entière.

La description est subdivisée en 3 parties qui reflètent des niveaux de détails croissants : description synthétique, analyse globale et analyse détaillée.

6.2. DESCRIPTION SYNTHETIQUE.

A l'aide du vocabulaire et de l'environnement définis ci-dessus, il est maintenant possible de donner une description synthétique, en langage courant, de l'algorithme qui va être analysé de façon de plus en plus détaillée dans la suite :

Etant donnés . "spp" OTUs
. et les états des "chars" variables relatifs à ces OTUs
. et les états primitifs de ces variables
. et les types évolutifs de ces mêmes variables
. (et éventuellement une option émanant de l'utilisateur et imposant un ordre aléatoire de prise en compte des OTUs)

utiliser une stratégie HEURISTIQUE

pour construire une liste d'arbres phylogénétiques
dont les spp OTUs sont les feuilles
et qui soient des arbres binaires comportant 2spp-1 sommets
et qui soient minima ou correspondant à des arbres ternaires minima [avec 2(spp+1)-2 sommets].

Remarques : 1.Dans l'algorithme de MIX, il faut bien distinguer le processus de construction qui concerne uniquement des arbres binaires (tout comme d'ailleurs la mémorisation) et le processus d'évaluation de la longueur de l'arbre (cf. procédure evaluate) qui est relative à l'arbre binaire construit ou à un arbre ternaire qui peut en être déduit (en y ajoutant une racine cladistique distincte de la racine binaire de construction -cf. fig. 6.4).

2.L'enracinement cladistique éventuel par outgroup n'intervient qu'aux stades de mémorisation et d'impression et n'est donc pas pris en compte ici.

6.3. ANALYSE GLOBALE.

Cette analyse reste à un niveau très conceptuel et a pour but de permettre une compréhension globale du processus.

6.3.1. **Description de l'algorithme en pseudolangage.**

6.3.1.1. Etape I : choix d 'un ordre de prise en compte des OTUs et construction d'un arbre binaire le plus parcimonieux possible (ou correspondant à un arbre ternaire le plus parcimonieux possible).

#1. Donner à chaque sommet-OTU un numéro d'ordre i de prise en compte par l'algorithme de construction (1≤i≤spp); (cf. p.III 21-22)

NB. 1.Cet ordre de prise en compte peut être celui de l'input (option par défaut) ou un mélange aléatoire de l'ordre d'input (option J).
2.Par convention, un sommet est désigné par son numéro (d'ordre d'input ou de prise en compte).Par convention également le numéro d'ordre de prise en compte est souligné tandis que le numéro d'ordre d'input ,non.

#2. Amorcer la construction de l'arbre binaire (cf. description de la procédure add) de façon à obtenir le résultat suivant :

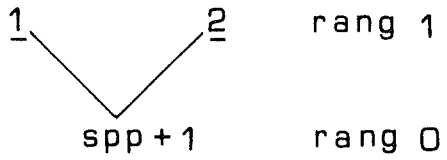


Fig.6.1.:Amorce de construction de l'arbre binaire.

et désigner par root la racine de l'arbre binaire;
(cf. p. III.22)

#3. Réaliser la sous-étape d'addition séquentielle-réarrangements locaux (cf. p. III.22) c.-à-d. :

pour i := 3 à spp

-ajouter le sommet-OTU i et son père (spp+i-1) à la localisation qui minimise ** la longueur de l'arbre (binaire ou ternaire) (cf. description des procédures addpreorder et add), en continuant à désigner par root la racine de l'arbre binaire;

-répéter le traitement suivant jusqu'à ce qu'il n'apporte plus de diminution de la longueur de l'arbre :

* Effectuer un parcours préordre de l'arbre binaire en essayant un réarrangement local (voir fig. 6.5) de l'arbre à partir de chaque sommet. A chacun de ces essais :
  si l'arbre ainsi obtenu est strictement plus court que l'arbre construit jusqu'alors : le conserver
  sinon : restaurer l'arbre original;

(cf. description de la procédure rearrange).

**NB: S'il y a plusieurs sommets à partir duquel l'ajout fournit un arbre de parcimonie maximale, alors l'ajout effectif est à faire à partir du premier des sommets rencontré dans un parcours préordre de l'arbre).

6.3.1.2. Etape II: Réarrangements globaux. (cf. p. III.22)

-Répéter le traitement suivant jusqu'à ce qu'il n'apporte plus de diminution de la longueur de l'arbre :

.pour j := 1 à nonode
(tout en veillant à continuer à désigner par root la racine de l'arbre binaire,)
si le sommet j est ≠ de root :
* enlever le sommet j (restant lié à tous ses éventuels descendants) et son père de l'arbre binaire; (cf. description de la procédure remove)
* ajouter temporairement ce sommet j (restant lié à tous ses éventuels descendants) et son père successivement à chacun des sommets de l'arbre (selon un parcours préordre). Ce faisant, mémoriser chaque arbre binaire complété ainsi obtenu dont la parcimonie est au moins égale à celle de l'arbre complété le plus parcimonieux jusqu'alors. (cf. description de la procédure addpreorder)
* ajouter finalement ce même sommet j (restant lié à tous ses éventuels descendants) et son père à la localisation ** qui minimise la longueur de l'arbre. (cf. description de la procédure add).

Remarques : 1. Le processus de réarrangements globaux se rapproche davantage de celui de l'addition séquentielle que de celui des réarrangements locaux !

2. Pour plus de clarté et d'élégance informatiques, l'étape de réarrangements globaux qui se réalise lorsqu'un arbre binaire complété a été obtenu, devrait être retirée de la boucle de construction de l'arbre (for i := 3 to spp ...) (voir p. III.22. Cette étape forme une entité et pourrait donc former une procédure appelée par maketree.

** NB: Voir note en fin de page précédente

6.3.2. Spécifications sous forme d'assertions des deux grandes étapes de l'algorithme de recherche.

6.3.2.1. Etape I:

Arguments : spp : entier
nonode: entier
nonode sommets

préconditions :

spp : est le nombre d'OTUs fournis pour l'analyse par l'utilisateur.

nonode : est le nombre de sommets dans l'arbre binaire correspondant à spp OTUs.
(nonode = 2spp-1)

les sommets sont de degré 0.

les sommets-OTUs sont identifiés par un entier de 1 à spp (correspondant à l'ordre d'input).

les sommets-ancêtres sont identifiés par un entier de spp+1 à nonode.

Résultats : AB : arbre binaire
root : sommet
bestyet : entier

postconditions :

AB est . complète

. parcimonieux* (ou correspond à un arbre ternaire parcimonieux)

---

* [En effet, il maximise **localement** le critère de parcimonie : c.-à-d. qu' il est au moins aussi parcimonieux que tout autre arbre différant de lui par de simples réarrangements locaux . Cependant, il ne répond pas nécessairement au critère de parcimonie globale et, à fortiori, n'est pas nécessairement le plus parcimonieux des arbres (cf. commentaires de Felsenstein dans MAIN.DOC)]

. enraciné ou non-enraciné* cladistiquement, selon les cas.

root : désigne la racine de AB

bestyet : = (le nombre de changements évolutifs impliqués par AB ou par l'arbre ternaire correspondant).

---

* [Il est non enraciné cladistiquement si et seulement si toutes les variables sont de type Wagner et sont d'état primitif "inconnu".
Remarques :
1.Ceci correspond à l'option par défaut de MIX.
2.Ce cas de figure est également obtenu lorsque l'utilisateur utilise l'option A seule et donne lui-même la valeur "inconnu" à l'état primitif de chaque variable , tout en conservant toutes les autres options par défaut.
3.Pour rappel :
l'option O d'enracinement sur base d'un outgroup désigné par l'utilisateur n'intervient qu'au moment de la mémorisation (cf. procédure savetree, p. III.12) et de l'impression de l'arbre. Ce mode d'enracinement cladistique sera donc qualifié de secondaire.]

[Il est enraciné cladistiquement par construction dans tous les autres cas, c.-à-d. | si au moins une variable de type Wagner a son état primitif (0 ou 1) désigné par l'utilisateur (option A) ou
| si au moins une variable est de type Camin-Sokal (même si son état primtif est déclaré inconnu)
L'enracinement cladistique par construction sera ici nommé Primaire pour l'opposer à l'enracinement secondaire décrit ci-dessus.
Remarque : Cette dernière condition suffisante à l'enracinement cladistique que l'on trouve dans le code PASCAL du programme MIX est signalée dans MIX-DOC : "Note thant when any of the characters has Camin-Sokal parsimony assumed for it, the tree is rooted..." mais elle paraît en contradiction avec l'affirmation de Felsenstein dans sa synthèse de 1983 (p. 317):
"It is an interesting fact that this 'unordered' variant of Camin-Sokal parsimony does not necessarily result in an unrooted phylogeny."]

6.3.2.2. Etape II.

Arguments : AB : arbre binaire

root : sommet

bestyet : entier

préconditions : voir les postconditions de l'étape I

Résultats : bestrees : vecteur de [1...100] de vecteurs de [1...100] d'entiers

bestlike : entier

postconditions :

. la topologie d'un arbre binaire sélectionné pour l'output est mémorisée dans bestrees.

[NB : les modalités de cette mémorisation sont volontairement passées sous silence ici.]

. 1 arbre binaire sélectionné
- est le fruit de réarrangements globaux dont le point de départ est AB
- a (ou correspond à un arbre ternaire ayant) une parcimonie à besyet et à bestlike.

Remarque : le(s) meilleur(s) arbre(s) trouvé(s) par l'algorithme de maketree est celui (sont ceux) dont la parcimonie = bestlike.

6.4. ANALYSE DETAILLEE

6.4.1. **Arbre des appels de la procédure "maketree"** (telle qu'elle est délimitée au paragraphe 6.1).

Cet arbre est consigné dans le tableau ci-dessous. Il faut noter que les procédures ne sont mentionnées qu'une fois par niveau même si elles y sont appelées plusieurs fois.

Tab.6.1. Arbre des appels de la procédure maketree

Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5	Niveau 6
add					
add-preorder	tryadd	savetree			
		findtree			
		addtree			
		add			
		evaluate	postorder	fillin	
				count	
			count		
		remove			
rearrange	repreorder	tryrearr	remove		
			add		
			evaluate	postorder	fillin
					count
				count	
remove					

6.4.2. **Description de différentes procédures appelées par maketree.**

Cette partie de l'analyse décrit et commente systématiquement et successivement chacune de ces procédures appelées par maketree, pour la recherche des différentes topologies les plus parcimonieuses. Toutefois elle n'entre pas dans les détails de l'implémentation.C'est ainsi qu'ont été passés sous silence le recours à l'utilisation de pointeurs et les conséquences des restrictions imposées par le langage Pascal concernant la taille du type de base des types ensembles. De même, la manière dont les données de l'input (cf.paragraphe 5.4.1.) sont implémentées en Pascal n'est pas précisée ici. Mais ces données sont supposées accessibles par la procédure maketree et n'ont pas été obligatoirement reprises dans les arguments des différentes procédures. D'autre part, d'une manière générale, la façon dont un arbre est représenté reste caché dans cette description.

Selon les cas, les procédures appelées par maketree sont ici plus ou moins détaillées. Seule une définition est donnée si la procédure effectue un traitement simple, immédiatement compréhensible à partir du code-source. Pour les procédures plus complexes, des spécifications sont données par assertions avec éventuellement des règles de traitement ou un algorithme en pseudolangage ou encore des schémas explicatifs.

En bref, cette troisième partie ne s'intéresse pas à la façon dont Felsenstein a formulé en Pascal son algorithme : elle dégage, pas à pas, la façon dont cet auteur applique le critère de parcimonie, selon les différents cas de figure, dans le cadre de l'analyse cladistique.

Procédure **add** (below, newtip, newfork)
[cf. fig. 6.2. et p. III.9].

Objectif : Ajouter un sommet (avec tous ses descendants éventuels) et un sommet-ancêtre à l'arbre binaire AB.

Arguments : below : sommet
newtip : sommet
newfork : sommet
AB : arbre binaire

préconditions : AB est non vide
below : sommet lié à AB
newtip : sommet non lié à AB, racine d'un arbre binaire ab
newfork: sommet isolé, ancêtre

Résultats : AB' : arbre binaire.

postconditions : newtip et newfork sont liés à AB' de la façon suivante :
. newfork a reçu,pour fils gauche, newtip (lui-même éventuellement lié à des descen - dants) et, pour fils droit, below.
. l'ancien père de below (s'il existait) est devenu l'actuel père de newfork.

Remarques : 1.A l'appel, below peut être de degré 0 (c'est le cas lors du 1er appel de add par maketree) (Dans ce cas-là, AB ne comporte que below).

2.Lors de la construction de l'arbre binaire (lorsque add(...) est appelé par addpreorder(...)) newtip sera un OTU (et deviendra par l'action de add une feuille de l'arbre binaire).

3.Après l'appel de add (below, newtip, newfork): newtip sera toujours fils gauche de newfork. Autrement dit, en appelant add (6, item, fork) pour la fig. 6.3.C , on ne pourra avoir que le schéma de la fig. 6.3.A (et jamais la fig. 6.3.B). (Bien sûr,ces 2 derniers schémas correspondent à une topologie unique).

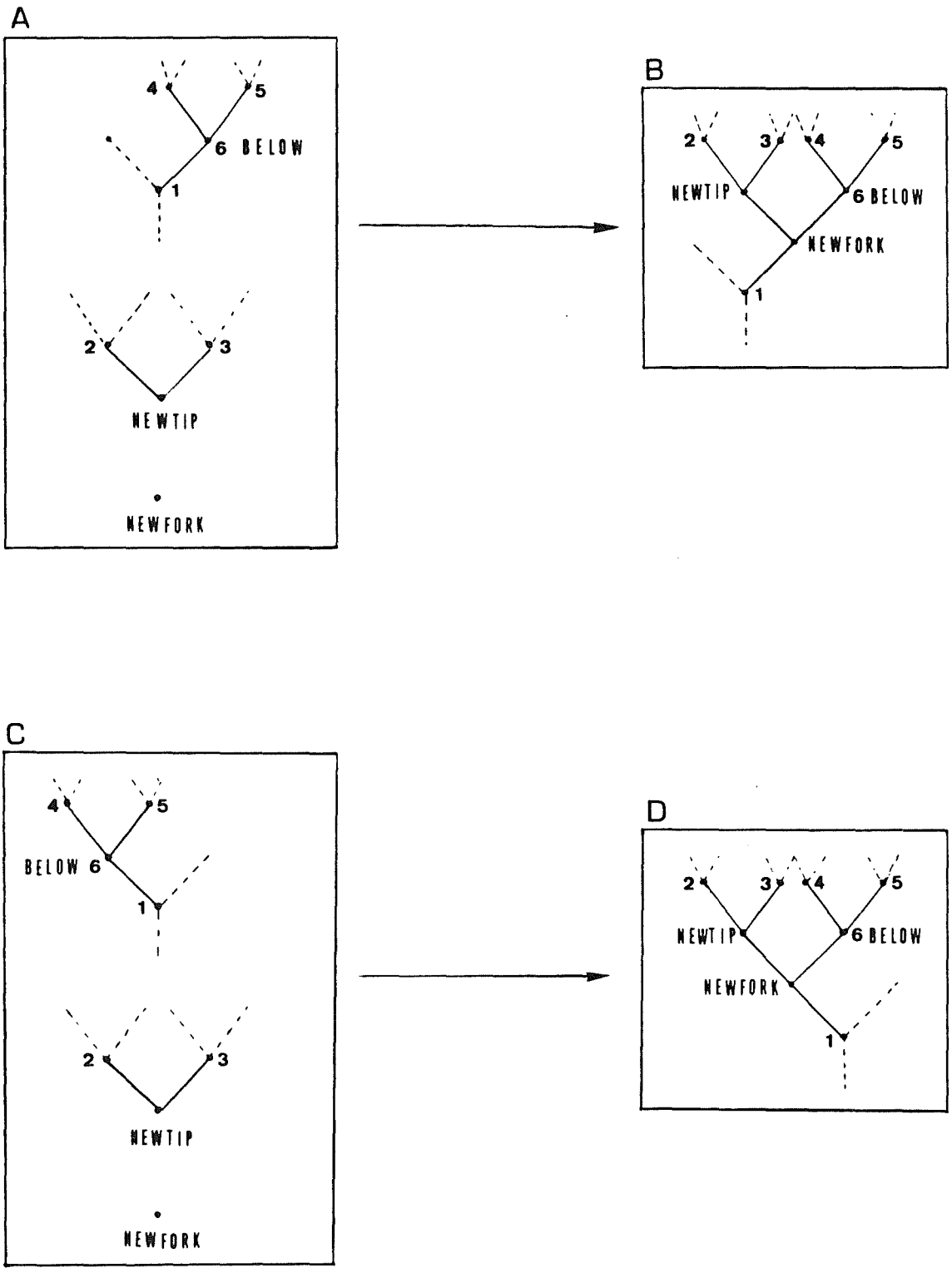


Fig. 6.2 Explication schématique des effets de l'application de la procédure add (below ,newtip ,newfork) sur 1 cas général :

au départ: below est de degré 3
newtip est de degré 2
newfork est nécessairement isolé.

N.B.: en A : below est fils droit.
en C : below est fils gauche.

Procédure **remove** (var item, fork)
[cf. fig. 6.3. et p. III.93

Objectif : Retirer de l'arbre binaire AB le sommet item (restant lié à tous ses éventuels descendants) et son sommet-père fork.

Arguments : AB : arbre binaire

item : sommet

fork : sommet

préconditions : item est lié à AB

fork est le père d'item

Résultats : AB' : arbre binaire

item : sommet

fork : sommet

postconditions : item (et tous ses descendants éventuels auxquels il reste lié) est détaché de AB'

fork est isolé

AB' est "cicatrisé", c.-à-d. que l'ancien père de fork est à présent lié à l'ancien fils (autre qu'item) de fork.

Remarques : 1. Lorsque remove est appelé en cours de construction de l'arbre, item est un OTU (et est donc de degré 1).
Le cas général où item est de degré >1 ne se présente qu'au cours des réarrangements (locaux et globaux)

2. Il faut noter qu'on obtient le même résultat, qu'item soit, au départ, fils gauche ou fils droit.

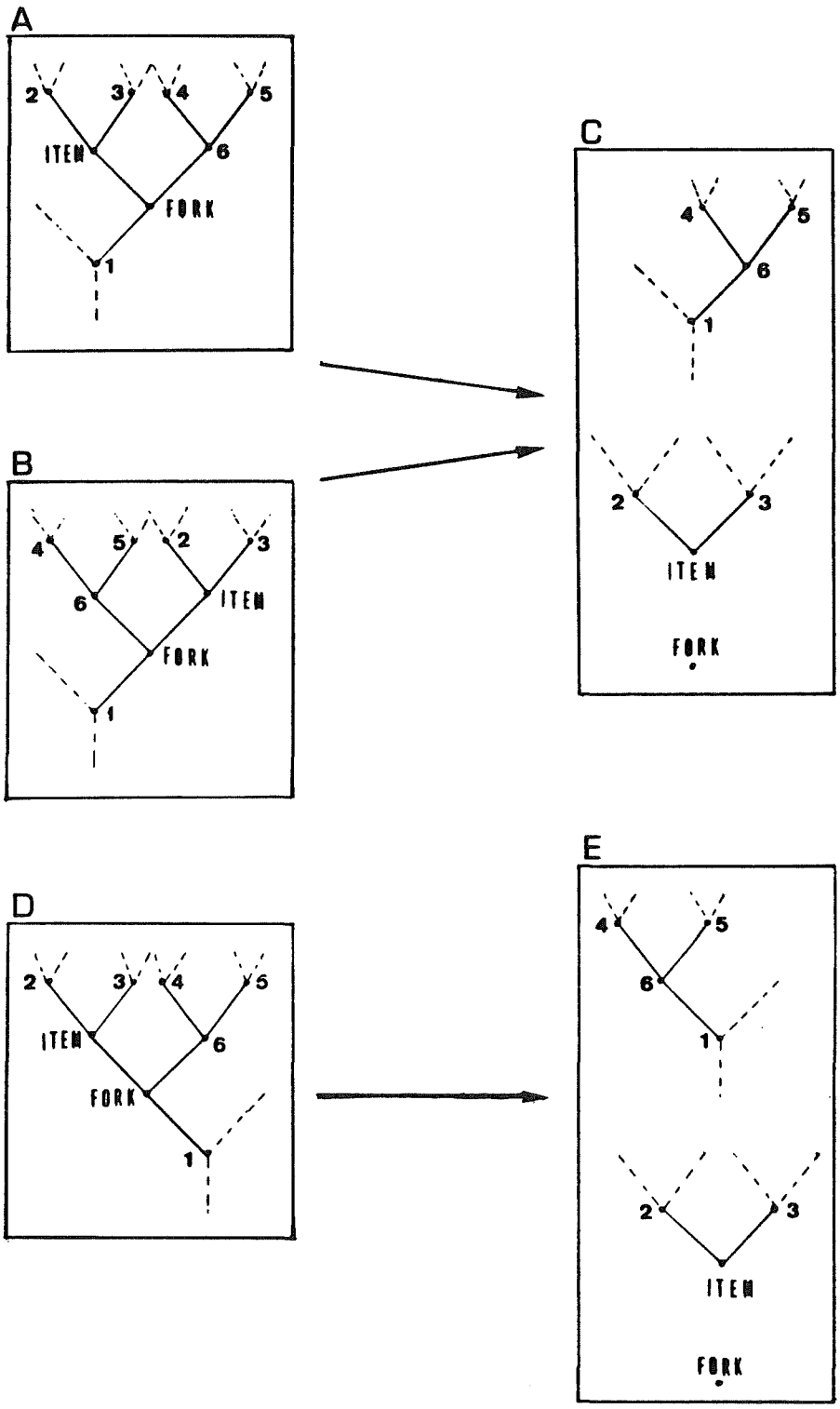


Fig. 6.3 Explication schématique des effets de l'application de la procédure remove (var item, fork) sur un cas général : au départ : item est de degré 3, fork aussi.
N.B. : en A :item est fils gauche et fork est fils droit,
en B :item est fils droit et fork est fils droit,
en D :fork est fils gauche.

Procédure **fillin** (p)
[cf. p. III.93]

Remarque : Afin de ne pas entrer dans des détails d'implémentations, un des arguments de cette procédure, ainsi que de la suivante, a été simplifié.

Objectifs: 1. Etant donné un noeud p : identifier les variables étant dans l'état 1 chez un fils, et dans l'état 0 chez l'autre fils de p.

2. Tenant compte de cette identification : déduire pour chaque variable, l'état au sommet p, conformément à son type évolutif et à l'hypothèse courante (imposée arbitrairement par le programme) concernant son état primitif.

Arguments : AB : arbre binaire
p : sommet
fg : sommet
fd : sommet
hyp : valeur de l'état primitif de toutes les variables telle qu'elle est imposée arbitrairement par le programme.

Remarque : "hyp" est une simplification de zeroanc[i] pour tout i de 1 à words tels qu'ils sont initialisés dans la procédure evaluate (voir ci-dessous) .

préconditions : fg,fd et p appartiennent à AB
p est un noeud
fg est fils gauche de p
fd est fils droit de p
hyp = 0 ou = 1
les états des variables sont déjà établis dans l'hypothèse courante, imposée arbitrairement concernant tous les états primitifs, pour fd et fg mais pas pour p.

Résultats : steps : ensemble des variables dont l'état est 1 chez un fils de p et 0 chez l'autre fils de p.
p : sommet.

postconditions : steps est évalué et
l'état de chaque variable est établi pour p selon la règle de traitement décrite au tableau 6.2.

TABLEAU 6.2
Règles de traitement pour l'établissement de l'état de chaque variable du noeud p dans la procédure fillin (p)

pour chaque variable :																				
si état au fg de p	1	0	1	0	1	0	1	0	?	?	?	0	1	P	1	P	0	P	?	P
si état au fd de p	1	0	0	1	0	1	0	1	?	1	0	?	?	1	P	0	P	?	P	P
si type évolutif=	-	-	W	W	CS	CS	CS	CS	-	-	-	-	-	-	-	-	-	-	-	-
si état primitif= (selon hypothèse arbitraire)	-	-	-	-	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-
alors état du noeud p =	1	0	?	?	1	1	0	0	?	1	0	0	1	1	1	0	0	?	?	?

Légende : ? = inconnue au non sens
P = polymorphe c-à-d. simultanément 0 et 1
W = Wagner
CS = Camin Sokal
- = quelle que soit la valeur

Remarques :

1. Steps sera utilisé comme paramètre actuel de count (voir ci-dessous) .

2. L'utilisateur doit être bien conscient de la règle de traitement par MIX de l'état inconnu d'un variable (cf.tableau 6.2.), de sa signification et de ses implications ! En effet cette règle consiste à faire l'hypothèse parcimonieuse qu'il n'y a pas de changement évolutif pour cette variable entre le père et ses 2 fils. Elle considère donc implicitement l'état "?" au niveau d'un fils dont le frère est 0 (/1) pour cette variable comme étant en réalité aussi 0(/1) !
   L'attitude de Felsenstein par rapport aux valeurs inconnues apparaît ici tout à fait analogue à celle que Swofford (1985,p.3.11) a adoptée dans PAUP : "... missing is equivalent to 'all possible states'. Although the algorithm does not work exactly in this way, the effect is that all missing states are 'filled in' according to what would be the most parsimonious character state had it not been missing, and the tree length then computed."

3. Un complément d'information concernant le traitement par MIX de polymorphisme et de l'état inconnu peut être trouvé dans la synthèse du paragraphe 6.5.

4. Les commentaires relatifs à cette procédure fillin devraient être mis à jour. "Sets up for each node" devrait être remplacé par "sets up for a node..." Quant à la phrase "if a character is in both sets it is in state "P", elle est peut-être un reliquat d'une version antérieure. Dans la version actuelle de MIX, elle est erronnée.

5 .La règle de traitement utilisée par la procédure fillin est plus contraignante que la règle de réécriture mentionnée à propos de l'exemple de Système Cladistique (cf. tab. 3.2) .Elle prend , en effet en compte le critère d'optimisation.
   De plus ,elle est plus simple puisque l'arbre à construire ici est binaire (et non ternaire comme dans l'exemple).

Procédure **count** (var stps)
[cf. p. III.10]

Objectif : Pour chacune des variables, séparément, mise à jour éventuelle de la somme cumulée du nombre de changements évolutifs déjà encourus.

Arguments : . stps : ensemble de variables
. numszero[1..chars] : tableau d'entier
(chars = le nombre de variables fournies par l'utilisateur).
. numsone[1..chars] : id.
. hyp : cf. arguments de la procédure fillin (p).

préconditions :. pour tout i de 1 à chars :
numszero[i] représente la somme cumulée des changements évolutifs déjà subis par la variable i dans l'hypothèse arbitraire où l'état primitif de la variable est 0.
numsone[i] ... est 1.
Remarque : l'initialisation à zéro de numszero[i] et de numsone[i] et réalisée par la procédure evaluate.
. hyp = 0 ou = 1.

Résultats : . numszero'[1..chars] : tableau d'entier
. numsone'[1..chars] : tableau d'entier

postcondition : pour tout i de 1 à chars, la mise à jour éventuelle de numszero[i] ou de numsone[i] est effectuée selon la règle décrite ci-dessous:

TABLEAU 6.3.:
règle de mise à jour éventuelle pour toute variable i de 1 à chars par la procédure count :

. variable i in stps	oui	oui	non
. hyp	0	1	-
numszero[i]	incrémenté de 1	inchangé	inchangé
numsone[i]	inchangé	incrémenté de 1	inchangé

Procédure **postorder** (p) : procédure récursive effectuant un parcours et un traitement postordres d'un arbre binaire. [cf. p. III.10]

Le traitement est le suivant :
[si p est une feuille : ne rien faire
[sinon appeler la procédure fillin(p)
puis la procédure count(steps).

Remarque : au premier appel de postorder le paramètre actuel sera toujours la racine (au sens de la théorie des graphes) de l'arbre binaire.

Procédure **evaluate** (r)
[cf. p. III.10]

I.Objectif : Evaluer la longueur de l'arbre c.-à-d. le nombre de changements évolutifs (-like) requis par toutes les variables dans un arbre binaire AB donné ou dans l'arbre ternaire correspondant à AB. ("+like" est appellé la parcimonie ou la vraisemblance de l'arbre par Felsenstein).

NB :Si l'état primitif 0 (ou 1) d'une variable i n'est pas imposé au départ (soit par l'utilisateur, soit par défaut) et si la contribution à like de cette variable varie selon l'état primitif : alors il faut choisir la contribution qui maximise like et mémoriser l'état primitif correspondant.

II.Spécifications:

Arguments : AB : arbre binaire.
r : sommet.

préconditions : AB est non vide.
r est la racine de AB.

Résultat : like : entier (<0).
guess : tableau [1..chars] de caractères
avec chars = le nombre de variables décrivant les OTUs

postconditions :
. (-like) : totalise les changements évolutifs de

toutes les variables dans le cadre d'une analyse de parcimonie
soit pour l'arbre ternaire obtenu en ajoutant la racine cladistique en dessous de la racine binaire (voir fig.6.4. et fig. 5.3) pour le cas où il y a un enracinement primaire de l'arbre et où la racine cladistique et la racine binaire sont distinctes;
soit pour l'arbre binaire AB dans tous les autres cas de figure.

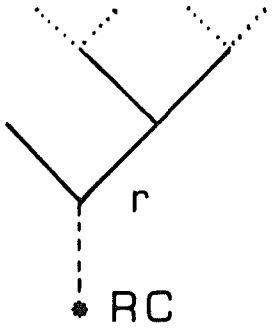


Fig.6.4.: Racine cladistique (RC) et racine (r) de l'arbre binaire.

. Pour tout i de 1 à chars:
guess(i)=
- 0 si et seulement si l'état primitif de la variable i vaut 0
  -soit selon l'indication de l'utilisateur;
  -soit par déduction du programme dans le cadre de l'analyse de parcimonie;
  -soit conformément à l'option par défaut relative aux variables de type Camin-Sokal.
- 1 si et seulement si l'état primitif de la variable i vaut 1
  -soit selon l'indication de l'utilisateur
  -soit par déduction du programme dans le cadre de l'analyse de parcimonie.
- ? si et seulement si l'état

primitif de la variable i
-n'est pas forcé à la valeur 0 ou à la valeur 1 par l'utilisateur ou par une option par défaut du programme
-et est indifférent.

Remarque : guess[i] sera imprimé uniquement dans le cas où l'arbre est enraciné cladistiquement.

III. Construction

Situation initiale :

```
sum := 0 ;
pour i := 1 à chars faire
        numszero[i] :=0 ;
            (NB: numszero[i]=somme cumulée des pas
                 évolutifs pour la variable i dans
                 l'hypothèse arbitraire que l'état
                 primitif de cette variable est zéro)

        numsone[i] :=0 ;
            (NB: numsone[i] = ...
                                  est 1.)
                fin faire
```

Séquencement des traitements :

```
#1.hyp := 0 ;
                (NB. hyp est la valeur de l'état
                     primitif pour toutes les varia-
                     bles, telle qu'elle est imposée
                     par hypothèse arbitraire (cf.
                     arguments des procédures fillin et
                     count.) )

  .appeler postorder (r) ;
                (NB. pour tout i, la valeur de
                     numszero[i] est maintenant
                     relative à la totalité de AB.)

  .appeler count (ensemble des variables valant 1 au
                     sommet r);
                (NB. Pour toute variable i valant 1
                     au sommet r, c.-à-d. à la racine
                     binaire de AB, la valeur de
                     numszero[i] est maintenant
                     relative à l' arbre ternaire
                     correspondant à AB.( Voir à ce
```

```
                        propos le paragraphe 6.5.3. et la
                        fig. 6.4 )

#2.hyp := 1;

  .appeler postorder (r);
                  (NB. Pour tout i, la valeur de
                       numsone[i] est maintenant
                       relative à la totalité de AB.)

  .appeler count (ensemble des variables valant 0 au
                  sommet r);
                  (NB. Pour toute variable i valant 0 au
                       sommet r, c.-à-d. à la racine
                       binaire de AB, la valeur de
                       numsone[i] est maintenant
                       relative à l'arbre ternaire
                       correspondant à AB.)


#3.pour i := 1 à chars faire

    * ┌ si l'état primitif de la variable i est connu
      │  et vaut 1, alors : . sum := sum + numsone[i];
      │                     . guess[i] := 1 ;
      │
      ├ si l'état primitif de la variable i est connu
      │  et vaut 0, alors : . sum := sum + numszero[i];
      │                     . guess[i] := 0 ;
      │
      └ sinon ┌ si numsone[i] < numszero[i],alors :
              │                . sum := sum + numsone[i];
              │                . guess[i] := 1 ;
              │
              └ sinon          . sum := sum + numszero[i];
                               ┌ si numszero[i]<numsone[i]
                               │      alors guess[i] := 0
                               └ sinon guess[i] := ? ;
                                   (NB.cas où numszero[i]=
                                        numsone[i] )

    * sum := sum + (nombre de fois que la variable i a
                    été déclarée F (ou B) dans la
                    matrice de description des OTUs);
                        (NB. Contribution des cas de
                             polymorphisme.)

      fin faire

#4.like := -sum.
```

(NB. Like définit selon Felsenstein la "vraisemblance" de l'arbre. Selon les cas, like est relatif à l'arbre binaire AB ou à l'arbre ternaire correspondant : voir post - conditions ci-dessus.)

IV. Remarques :

1. C'est l'appel à count (ensemble des variables valant 1 [/0] au sommet r) qui permet de tenir compte, le cas échéant, dans l'évaluation de la longueur, d'un sommet non implémenté et surnuméraire par rapport au 2spp-1 sommets de AB (cf. fig. 6.4).

2. L'étape # 3 dans le séquencement des traitements décrit ci-dessus est en fait "explosé" dans l'algorithme de Felsenstein, ce qui a pour conséquence une multiplication des tests et un allongement du code.

3. Rappel : les cas "usertree" et "threshold" de Felsenstein ne sont pas pris en compte ici.

4. Dans l'un de ses articles, Felsenstein (1983,p.317) fait une allusion implicite au séquencement des traitements exécuté par évaluate !

Procédure **tryadd** (p)
[cf. p. III.12]

Remarque préliminaire: La procédure tryadd (p) est appelée par addpreorder (p, item, nufork) au cours des 2 grandes étapes (cf. parag. 6.3) intervenant dans la procédure maketree. Dans la dernière étape, l'appel à addpreorder est précédé de l'appel de remove (item, nufork).

Objectifs:

- Ajouter temporairement un sommet (item) (avec tous ses éventuels descendants) et son sommet-père (nufork) à l'arbre binaire AB (non complété) à partir du sommet p lié à AB.

- Si l'ajout de ce sommet (item) est fait pour la première fois dans l'étape en question, ou si la localisation (p) de cet ajout donne un arbre binaire AB * strictement plus parcimonieux (ou correspondant à un arbre ternaire strictement plus parcimonieux) que les autres localisations de ce même ajout testées jusqu'alors (dans cette étape) :
alors mémoriser "p" dans "there"

- De plus, si en cours d'un réarrangement global, l'ajout à partir de p a pour résultat un arbre binaire AB* au moins aussi parcimonieux que l'arbre binaire complètement construit AB** le plus parcimonieux jusqu'alors :
alors mémoriser également la topologie de AB*
(! pour autant que cela ne soit pas déjà fait).

(NB. Au cas où l'évaluation de la parcimonie est relative à un arbre ternaire correspondant à l'arbre binaire construit, l'énoncé de cet objectif est : de plus, si ... a pour résultat un arbre binaire AB* correspondant à un arbre ternaire AT* au moins aussi parcimonieux que l'arbre ternaire AT** correspondant à l'arbre binaire AB** complété le plus parcimonieux jusqu'alors ... fait).

Arguments: AB : arbre binaire.
p : sommet.
root : sommet.
item : sommet.
nufork : sommet.
there : sommet.
bestyet : entier.
bestlike : entier (NB : n'intervient qu'à l'étape II)
lastrearr : booléen.

Remarque : les arguments des procédures savetree, findtree et addtree sont volontairement omis ici.

préconditions :

AB est non-vide et non-complété.
root est la racine de AB.
p et there sont liés à AB.
item, racine de l'arbre binaire ab, est non lié à AB.
nufork est un sommet-ancêtre isolé.
bestyet <0 (!initialisé à -32000).
bestlike<0 ("parcimonie" c.-à-d. (-longueur) de l'arbre binaire complété ou de l'arbre

ternaire correspondant actuellement le plus parcimonieux).

lastrearr = soit false
et l'étape en cours est celle de la construction de l'arbre binaire (et bestlike n'intervient pas) ;
soit true
et l'étape en cours est celle de réarrangements globaux (et bestlike intervient).

Résultats: AB arbre binaire
root : sommet
there' : sommet
item : sommet
nufork : sommet
bestyet' : entier
bestlike' : entier
like : entier
bestrees : vecteurs de [1..100] de vecteurs de [1..100] d'entiers

Remarque : les résultats des procédures savetree, findtree et addtree sont volontairement non explicités ici.

postconditions :

AB-résultat est identique à AB-argument puisque l'ajout a été temporaire.
De même, les préconditions de root, item et nufork sont aussi leurs postconditions);
bestyet' et bestlike' et like<0;
like est la "parcimonie" c.-à-d. (-longueur) de l'arbre AB* (ou de l'arbre ternaire correspondant) avec ajout;
bestyet' est la parcimonie c.-à-d. (-longueur) du meilleur arbre (binaire ou ternaire correspondant) obtenu jusqu'alors dans cette étape (et au cours de cette itération de la boucle for) lors de l'ajout d'item et de nufork.

┌soit bestyet' = bestyet
│ et bestyet'≥ like
│ et there' = there
└soit bestyet' > bestyet
et bestyet' = like
et there' = p ≠ there

si lastrearr = true
┌soit bestlike'= like > bestlike
│ et alors l'arbre AB* (c.à-d. avec ajout) a été mémorisé dans bestrees
└soit bestlike' = bestlike

alors si bestlike = like alors l'arbre AB*
a été mémorisé s'il ne
l ' était pas encore.

Remarque : voir remarque 2 de addpreorder.

Remarques postliminaires.

1. La procédure tryadd est complexe. Elle est criticable car elle est très hétérogène.
   Il est , d'autre part impossible de la comprendre sans devoir comprendre simultanément le mécanisme de construction de l'arbre et des réarrangements globaux (cf. maketree).

2. Le commentaire accompagnant cette procédure dans MIX.PAS mérite d'être précisé et complété .

Procédure **Addpreorder** (p, item, nufork) : procédure (récursive) effectuant un parcours et un traitement préordre de l'arbre binaire AB. [Cf. p. III.12]

Le traitement est le suivant : tryadd (p).

Remarque : 1. A l'appel, le paramètre actuel de p sera toujours root (c.-à-d. la racine binaire).

2. L'appel à addpreorder (p, item, nufork) est logiquement suivi de l'appel à add (there, item, nufork).

Procédure **tryrearr** (p).
[cf. fig. 6.5 et p. III.14]

Objectif: essayer un réarrangement local de l'arbre binaire AB à partir du sommet p lié à AB.

- Si cet essai est fructueux, c'est-à-dire si l'arbre AB' ainsi obtenu est strictement plus parcimonieux qu ' AB
  ou si l'arbre ternaire AT' correspondant à AB' ainsi obtenu est strictement plus parcimonieux que l'arbre ternaire AT correspondant à AB
  alors conserver AB'.
- sinon : restaurer l'arbre original AB

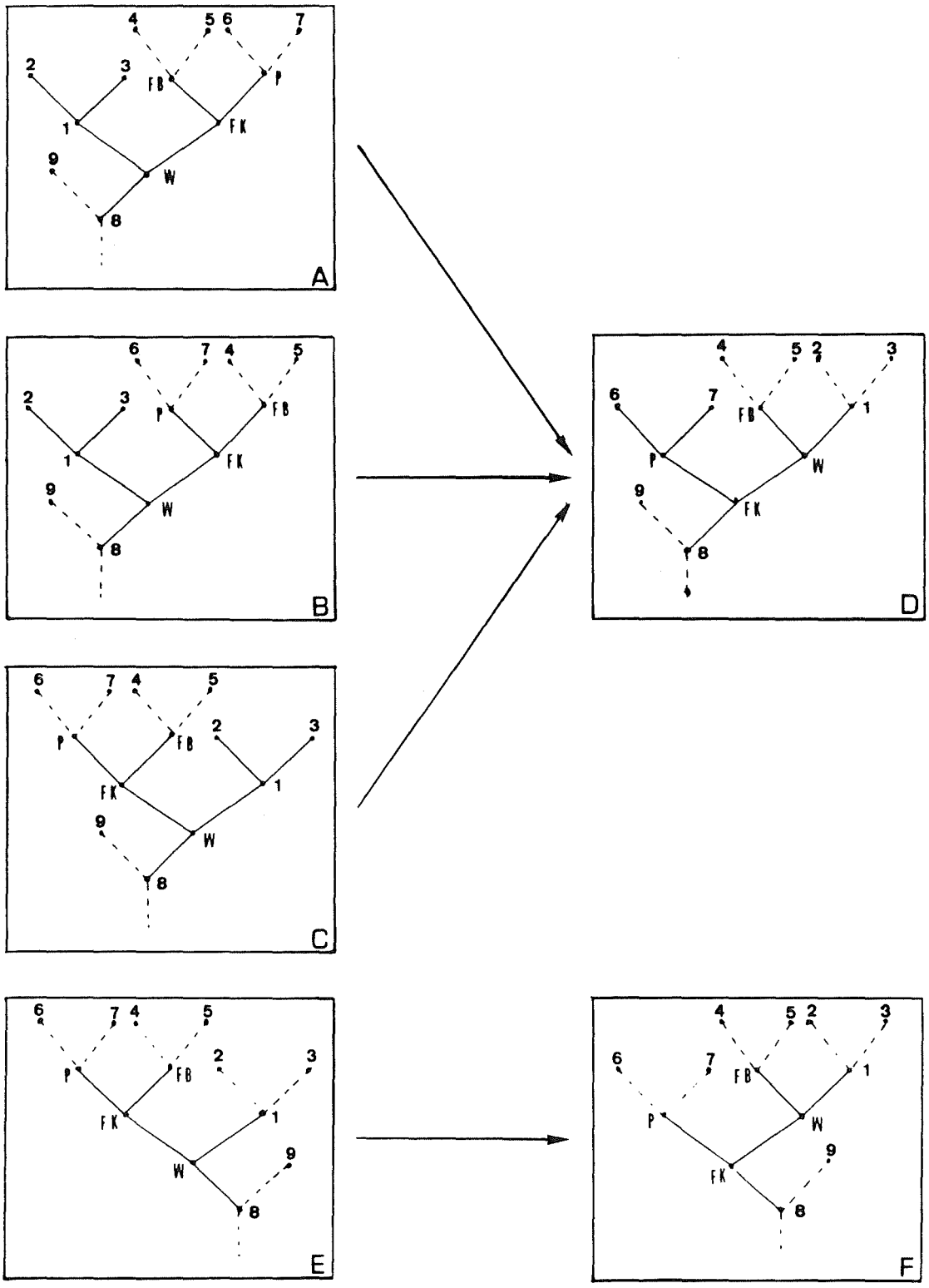


Fig.6.5 Explication schématique d'un réarrangement local résultant de la séquence d'intructions [remove (p,forknode); add (whereto,p,forknode);] contenue dans la procédure tryrearr (p) , dans un cas général où le degré de p, de son père et son grand-père = 3.
FK = forknode ,FB = frombelow ,W = whereto.

Arguments: AB : arbre binaire.
p : sommet.
r : sommet.
bestyet : entier.
success : booléen.

préconditions :
p est lié à AB et est de rang n.
bestyet <0 (="-longueur" de AB ou de l'arbre ternaire correspondant).
r est la racine de AB.

Résultats: AB' : arbre binaire.
p : sommet.
r' : sommet.
bestyet' : entier.
success' : booléen.

postconditions : p est lié à AB' et r' est racine de AB'

. soit AB' = AB
et p est de rang n
et bestyet' = bestyet
et success' = success
et r' = r.

. soit AB'≠ AB
et p est de rang n-1 et est fils gauche.
Il a gardé le même père mais celui-ci est maintenant de rang n-2.
Par contre , l'ancien grand-père de p est maintenant de rang n-1.
Quant à l'ancien frère de p, il est resté de rang n et est donc maintenant le fils de son ancien grand-père.
(NB : voir explication schématique : fig.6.5).
et bestyet' >bestyet (bestyet' est la parcimonie (c.-à-d. [- longueur]) de AB' (ou de l'arbre ternaire correspondant).
et success' = true
et soit p est de rang >1 (cf. p.I.2) et r' = r,
soit p est de rang 1 et r' est le père de p; r'≠r.

Remarques: 1.Les conditions d'essai d'un réarrangement local (à savoir le fait que p ait un père et un grand-père) sont testées à l'intérieur de la procédure tryrearr. Il aurait été plus "joli" de faire ce test en amont de l'appel de tryrearr !

2.Le résultat d'un réarrangement local ne change pas selon que p ,au départ , soit fils droit (cf.

fig. 6.5.A) ou fils gauche (cf. fig. 6.5.B). Il est également indépendant du fait que forknode ,initialement, soit fils droit (cf. fig. 6.5.A & 6.5.B) ou fils gauche (cf. fig 6.5.C).
La figure 6.5.D. sera donc le résultat unique de 4 schémas initiaux différents.(Ces derniers correspondent d'ailleurs à une topologie unique).
(Voir à ce propos la distinction entre sémantique procédurale et déclarative dans les annexes I et II.)
La même remarque s'applique pour la figure 6.5.F.
(De plus, ces deux figures [D & F] sont des représentations graphiques différentes d'une même topologie).

3. Un réarrangement local, tel qu'il est représenté dans l'explication schématique de la fig. 6.5 correspond d'un point de vue toplogique à la description générale des réarrangements locaux telle qu'elle est faite dans la documentation générale (MAIN.DOC, paragraphe "The algorithm for constructing trees") par Felsenstein : "A local rearrangement involves an internal segment of the tree in the following manner. Each internal segment of the tree is of the form (where T1, T2 and T3 are subtrees...)

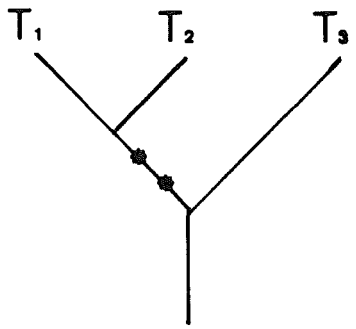


Fig. 6.6. a)
(extraite de MAIN.DOC de Felsenstein.)

the segment we are discussing being indicated by the asterisks. A local rearrangement consists of switching the subtrees T1 and T3 or T2 and T3 so as to obtain one of the following :

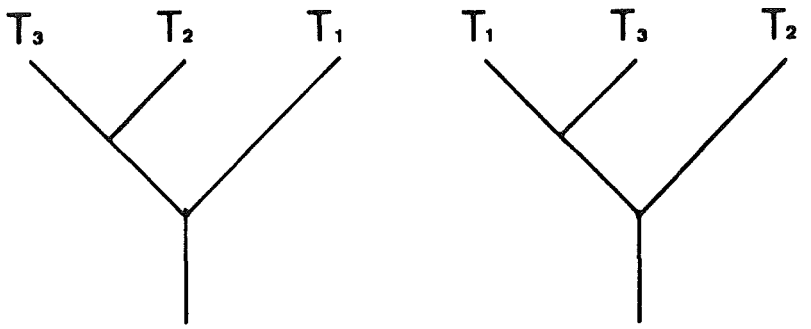


Fig. 6.6. b) & c)
(adaptée de MAIN.DOC de Felsenstein.)
Illustration d'un réarrangement local fournie dans la documentation de Felsenstein.

Pour comprendre la correspondance entre les graphiques de la figure 6.5 et ceux de Felsenstein (figure 6.6), il faut simplement réaliser que pour le premier réarrangement p vaut T1 tandis que pour le second, il vaut T2 et que le "internal segment" de Felsenstein est en fait l'arête joignant forknode à whereto. Il y a simplement un éclairage différent dans la documentation de Felsenstein où les réarrangements locaux sont décrits en prenant comme référence cet "internal segment" alors qu'ils sont réalisés dans MIX à partir d'un noeud p devant avoir un père et un grand-père.
Le réarrangement local est aussi conforme à la définition relativement vague qu'en donnent Luckow et Pimentel (1985, p. 49) : "...local branch swapping rearranges terminal tree within subtrees".

Procédure **repreorder** (p) : procédure récursive effectuant un parcours et un traitement préordre d'un arbre binaire. [Cf. p. III.15]

Le traitement est le suivant : tryrearr (p).

Remarque : Au premier appel le paramètre actuel de p serait toujours la racine de l'arbre binaire.

Procédure **rearrange** (var r) : procédure répétant l'appel de "repreorder (r)" jusqu'à ce qu'au cours d'un parcours (effectué par cette procédure) aucun réarrangement local ne réussisse plus : (c.-à-d. qu'on achève le parcours avec success = false). (cf. p. III.15)

Remarque : Le paramètre actuel de r à l'appel est la racine de l'arbre binaire.

6.5. REMARQUES SYNTHETIQUES.

6.5.1. **Le traitement du polymorphisme (P/B) par MIX.**

6.5.1.1. Comme signalé précédemment (cf.parag. 5.4.1.), le polymorphisme n'est pas admis par MIX comme état primitif de variable. (Si l'utilisateur introduit P comme état primitif, le programme le transforme automatiquement en ?).

6.5.1.2. L'utilisateur peut introduire le polymorphisme comme état de variable des OTUs analysés.

6.5.1.3. Par hypothèse, la rétention du polymorphisme est un évènement beaucoup moins probable que les changements 0/1 (cf. MIX-DOC). (Ceci est illustré par la règle du tableau 6.2.)

6.5.1.4. L'algorithme de construction de l'arbre de maketree fonctionne de la manière suivante :

a. il ajoute au total des changements évolutifs pour l'arbre (c.-à-d. à la longueur de l'arbre) autant d'unité qu'il y a d'états de variables déclarés polymorphes au niveau des OTUs (cf. procédure evaluate). Cet ajout est donc automatique et indépendant du critère de parcimonie à maximiser !

b. il se contente alors de traiter les états polymorphes comme s'ils étaient inconnus (cf. procédures fillin et count).

6.5.1.5. Ces règles de traitements ne sont pas formulées explicitement par Felsenstein , dans sa documentation. Ceci a pour conséquence que la compréhension du décompte des pas évolutifs imputables au polymorphisme et de leur localisation dans l'arbre peut ne pas être évidente pour l'utilisateur de MIX ... (cf. fig. IV.1)

6.5.1.6. Le traitement des états polymorphes par MIX n'a rien à voir avec la parcimonie de type polymorphisme ("Polymorphism parsimony") que Felsenstein décrit dans ses articles (1983, p. 318 et 319,1979 & 1982) en les comparant à d'autres types (Wagner sensu stricto, Camin-Sokal, ...).

6.5.2. **L'état inconnu ("?")**

6.5.2.1. L'utilisateur peut donner la valeur inconnue ("?") à un état d'une variable d'un OTU ou à l'état primitif d'une variable dans 2 cas de figures :

soit si la donnée n'est pas disponible (par exemple en raison du mauvais état de conservation pour un fossile)

soit parce que le caractère n'est pas d'application (par exemple la couleur blanche d'un appendice pour un OTU dépourvu de cet appendice).

6.5.2.2. La fréquence des données manquantes constitue un problème crucial en paléontologie. Fort heureusement les programmes récents d'analyse de parcimonie sont capables de s'en accomoder. Mais curieusement, on ne trouve dans la littérature cladistique que très peu d'informations concernant le traitement de l'état inconnu.
La documentation accompagnant le programme PHYLIP ne fait pas exception à cette règle: absolument rien n'y filtre concernant le "mode d'emploi" par l'algorithme de MIX des "?".

6.5.2.3. L'utilisateur doit être bien conscient que s'il donne la valeur "?" à un état de variable ou à un état primitif de variable , cela signifie qu'il passe tout à fait la main au programme à ce propos : selon la solution la plus parcimonieuse, MIX pourra remplacer ce "?" par 0 ou par 1 pour le calcul de la longueur d'un arbre (voir la procédure fillin). D'autre part, une méconnaissance de ce fait pourra rendre l'output très ambigu. (cf. fig. IV.2)

6.5.2.4 Il existe un glissement de signification du "?" pour un état de variable selon qu'il est relatif, d'une part,à un HTU dans l'output ou,d'autre part,à un HTU au niveau de la procédure fillin ou à un OTU .
En effet, si le programme imprime la valeur "?" à une variable d'HTU , cela signifie cette fois non pas que l'état de la variable est inconnu au niveau de ce HTU mais qu'il est aussi parcimonieux de lui attribuer la valeur 0 que la valeur 1 pour une topologie donnée.
Ceci est d'ailleurs clairement expliqué par MIX-DOC:
" If the inferred state is a "?", there will be multiple equally-parsimonious assignements of states; the user must work there out for themselves by hand."

6.5.2.5. Ce m@me glissement de sens se retrouve au niveau du "?" comme état primitif : à l'input,"?" signifie inconnu et non-imposé au départ. (Le programme fera tout à tour l'hypothèse que l'état primitif est 0 et puis qu'il est 1 pour rechercher la solution la plus parcimonieuse). A l'output (cf. valeur de guess calculée dans la procédure evaluate et de "guesses" dans l'"output"), le ? signifie que la solution où l'état primitif vaut 1 est aussi parcimonieuse que celle où il vaut 0. (cf. par exemple la fig. IV.2).

6.5.3. **Types de racines et types d'arbres.**

Le concept de racine est particulièrement ambigu et problématique en ce qui concerne le programme MIX. La racine à laquelle fait allusion la documentation destinée à l'utilisateur (cf. MAIN.DOC, DISCRETE.DOC et MIX.DOC) est la racine cladistique. C'est également cette racine qui désigne le mot "root" lorsqu'il apparaît dans l'output de MIX. Par contre, le terme "root" qui se trouve dans le texte-source en Pascal (MIX.PAS) n'est pas cette racine cladistique : il est relatif à la racine de l'arbre binaire(résultant du processus de construction) telle qu'elle est définie dans le vocabulaire emprunté à la théorie des graphes . Mais ceci n'est pas explicité par Felsenstein dans les commentaires de ses procédures.

Cependant, la racine par rapport au sens de construction de l'arbre est à bien distinguer de la racine cladistique (c.-à-d. racine par rapport au sens de l'évolution de la vie) dans l'algorithme de MIX (cf. fig. 5.3). Bien plus,la racine cladistique, si elle existe, peut avoir différents degrés selon les différents cas de figures imposés par l'utilisateur à l'input.

Il est éclairant d'élargir l'analyse et de souligner (cf. tableau 6.4.) non seulement les différents types de racines (cladistiques ou de construction; de degré 1 ou 2) mais aussi les différents types d'arbres complétés (binaires ou ternaires) impliqués par certaines phases de MIX (construction, évaluation de la parcimonie, impression de l'ouput). Il est également significatif de faire à ce propos la différence entre le niveau du programme et la réalité sous-jacente.

Pour aborder cette synthèse, il convient d'avoir bien présente à l'esprit la convention suivante imposée par MIX :
l'ancêtre hypothétique commun éventuel (ou racine cladistique) ne peut être compté parmi les spp OTUs analysés !
D'autre part, il faut accepter de faire l'hypothèse simplificatrice que tous les noeuds autres que la racine sont de degré 3 dans la réalité sous-jacente à l'évaluation (une arête sera donc comptabilisée même si elle est de longueur nulle).
Enfin, les conditions du non-enracinement cladistique et celles d'enracinement cladistique primaire et secondaires ne sont pas reprises ici. Elles peuvent être trouvées au paragraphe 6.3.2.1).

Le tableau synthétique (6.4) mérite de nombreux commentaires. En effet, il indique tout d'abord que l'algorithme de MIX construit **toujours** un arbre binaire comportant 2spp-1 sommets implémentés parmi lesquels un noeud de degré 2 (la racine binaire de construction).

En ce qui concerne l'évaluation de la parcimonie (avant mémorisation éventuelle), la situation est plus diversifiée et complexe. En effet, comme cela a été décrit ci-dessus, la procédure evaluate opère systématiquement, dans l'hypothèse arbitraire où toutes les variables ont 0 comme état primitif, un appel à count (ensemble des variables valant 1 à la racine r de construction de l'arbre binaire construit).
Ensuite, dans l'hypothèse arbitraire où toutes les variables ont 1 comme état primitif, evaluate fait appel à count (ensemble des variables valant 0 à la racine r de construction de l'arbre binaire construit). Tout se passe donc comme si sous cette racine binaire, pouvait exister un sommet supplémentaire et non implémenté correspondant à la racine cladistique et pouvant, en quelque sorte, être fictivement rattaché à l'arbre binaire construit pour obtenir, lors de l'évaluation de la longueur de l'arbre, un arbre ternaire avec 2(spp**+1**)-2 sommets. C'est ce qu'illustre la figure 6.4.

TABLEAU 6.4. : SYNTHESE : Types de racines et type d'arbres complétés, impliqués par certaines étapes de MIX

	RESULTAT DU PROCESSUS DE CONSTRUCTION	EVALUATION DE LA PARCIMONIE (avant mémorisation éventuelle)		OUTPUT
		utilisation effective par l'algorithme	réalité sous-jacente à cette évaluation	
I.CAS SANS ENRACINEMENT CLADISTIQUE	AB:[2spp-1]S	AB:[2spp-1]S sans RC	AT:[2spp-2]S sans RC	AB:[2spp-1]S
II.CAS AVEC ENRACINEMENT CLADISTIQUE SECONDAIRE	AB:[2spp-1]S	AB:[2spp-1]S sans RC	AT:[2spp-2]S sans RC	AB:[2spp-1]S
III.CAS AVEC ENRACINEMENT CLADISTIQUE PRIMAIRE en général	AB:[2spp-1]S	AB:[2spp-1]S RC:de degré=2	AT:[2spp-1]S RC:de degré=2	±AT:[2(spp+1)-2]S ±RC:de degré=1
cas particulier: RC ≠ racine de construction	AB:[2spp-1]S	AT:[2(spp+1)-2]S RC:de degré=1	AT:[2(spp+1)-2]S RC:de degré=1	±AT:[2(spp+1)-2]S ±RC:de degré=1

Légende: S = sommets
RC = racine cladistique
AB = arbre binaire
AT = arbre ternaire
spp = nombre d'espèce à analyser
NB.: Voir commentaires et détails supplémentaires dans le texte

La procédure evaluate calcule aveuglément ,pour toutes les variables, les longueurs dans chacune des deux hypothèses décrites ci-dessus, avant de retenir les longueurs pertinentes. Il se pourra donc que, même si l'arbre n'est pas enraciné cladistiquement de facon primaire, la racine cladistique intervienne quand même comme sommet supplémentaire au cours de l'algorithme d'evaluate! Mais dans ces cas -là (**sans enracinement cladistique ou avec enracinement cladistique secondaire** et donc non-pris en compte avant mémorisation), elle n'interviendra certainement pas dans la solution la plus parcimonieuse sélectionnée finalement par la procédure evaluate. Autrement dit, l'arbre évalué effectivement (cf. tableau 6.4) sera binaire et comportant 2spp-1 sommets. Mais ...Mais il faut bien comprendre qu'alors le fait d'avoir une racine binaire est un ARTIFICE DE CONSTRUCTION! car l'arbre est en réalité ternaire et comporte 2spp-2 sommets!
Au niveau de l'output (cf. paragraphe 5.4.2), MIX fournit un arbre binaire à 2spp-1 sommets aussi bien lorsqu'il n'y a pas d'enracinement cladistique (cf. fig. 5.2) que lorsqu' intervient secondairement l'enracinement cladistique (cf. fig. 5.4). Mais l'utilisateur s'apercevra après suppression de l'arête de longueur nulle (cf. paragraphe 5.4.2) que l'arbre non-enraciné cladistiquement est ternaire avec 2spp-2 sommets. Quant à l'arbre enraciné secondairement par outgroup, il acquiert, par rapport à l'arbre ternaire dont il est issu, un sommet supplémentaire (la racine cladistique) sur l'arête dont l'outgroup constitue la feuille (cf. paragraphe 3.2.0).
Il faut cependant remarquer que l'introduction de cette racine cladistique de degré =2 pourra aller de paire avec l'obtention d'une arête de longueur nulle !

En ce qui concerne un arbre avec **enracinement cladistique primaire**, il faut signaler un cas particulier intéressant : celui où la racine cladistique imposée (soit par défaut, soit par option A) ne coïncide pas avec la racine binaire de construction. Dans ce cas, l'évaluation de la parcimonie est effectivement relative à un arbre ternaire comportant 2 (spp+1)-2 sommets (cf. fig. 6.4) et la racine cladistique est de degré 1.
Dans tous les autres cas (et en particulier lorsque toutes les variables sont d'état primitif inconnu au départ et qu'au moins l'une d'entre elles est de type Camin-Sokal), l'évaluation de la longueur se fera sur un arbre binaire où racine cladistique et racine de construction seront confondues.
Au niveau de l'output, un arbre avec enracinement cladistique primaire (cf. fig. 5.3) est imprimé avec une arête supplémentaire issue de la racine de construction. Cet arbre peut donc être interprété comme un arbre ternaire muni d'une racine cladistique de degré 1 bien que ce dernier élément n'y figure pas.

A la lumière de cette analyse détaillée, il apparaît clairement que le fait de construire un arbre binaire est un choix d'implémentation fait par Felsenstein (voir à ce propos la comparaison avec Swofford au paragraphe 7.2.3.). Il faut remarquer que le fait de construire un arbre binaire permet une

simplification de l'algorithme : les règles de traitement pour l'établissement de l'état de chaque variable (cf. tableau 6.2) comportent un nombre restreint de clauses. En effet , tout sommet qui ne fait pas partie des spp OTUs a nécessairement deux fils. En revanche, des clauses plus nombreuse sont nécessaires pour construire un arbre ternaire, en utilisant un algorithme analogue à celui de MIX : il faut ,en effet tenir compte du cas particulier du sommet qui n'a qu'un fils (cf. tableau 3.2) .

Une dernière remarque doit être faite à propos du statut de la racine cladistique car il présente lui aussi une certaine ambiguité.
Si les états des variables (c.-à-d. les états primitifs) de cette racine sont déduits par programme, ce qui est le cas lors de l'enracinement cladistique par outgroup, alors la situation est claire : la racine cladistique est un HTU à part entière et n'appartient pas à l'ensemble X pour lequel les sommets dans f(X) sont appelés sommets réels (cf. vocabulaire relatif à la théorie des graphes).
En revanche, pour les états primitifs de caractères fixés à 0 ou 1 au départ (enracinement cladistique primaire), la racine cladistique se comporte comme un OTU.

6.5.4. Ordre de prise en compte pour l'algorithme et conséquence d'ordre stratégique.

Comme signalé au paragraphe 6.3.1.1,l'output de MIX dépendra de l'ordre de prise en compte des OTUs par l'algorithme. Ainsi donc, en faisant tourner plusieurs fois le programme et en variant chaque fois cet ordre (cf. option J et fig. 5.2), on multiplie les chances de trouver les arbres minimaux. Felsenstein (dans ses commentaires) et Platnick (1987) conseillent à l'utilisateur de faire ainsi au moins 10 "runs" sur les mêmes données. Ce fait pourrait être considéré comme une faiblesse du programme MIX. Felsenstein le présente cependant comme un avantage, jugeant que c'est un moyen facile de recherche.

CHAPITRE 7. COMPARAISONS.

7.1. INTRODUCTION.

Après avoir longuement analysé l'algorithme heuristique de MIX, il est intéressant de le situer parmi les autres programmes d'analyse phylogénétique (en particulier pour données discrètes) existants pour micro-ordinateur et d'effectuer des comparaisons concernant les possibilités offertes, tout d'abord ,et , ensuite, l'efficacité.

Il faut tout d'abord remarquer que MIX n'est qu'un des 30 programmes de PHYLIP 3.1 (PHYLogeny Inference Package) et que ce package présente une grande variété d'analyses : parcimonie de type Dollo et de type polymorphisme, compatibilité ...
Outre PHYLIP, il faut mentionner CLINCH écrit par Fiala et réalisant une analyse de compatibilité et MacClade de Maddison basé sur le principe de parcimonie (cf. Fink, 1986) mais surtout axé sur la reconstruction des états ancestraux de caractère pour une topologie donnée. Cependant le plus utilisé de tous ces programmes est , maintenant, semble-t-il, le PAUP de Swofford. La version distribuée en 1988 est la 2.4 qui est restreinte à la parcimonie de type Wagner (sensu stricto) mais étendue à des caractères non-ordonnés. Il existe une version ultérieure (2.99 J et K) déjà testée par Platnick (non encore publié) et présentant diverses améliorations par rapport à la version 2.4 : en particulier elle comprend, parait-il, une option "collapse" ayant pour effet d'éliminer les solutions redondantes. Un aperçu de la toute dernière version (3.0) de PAUP a été présenté à Stockholm en août 1988 au "Willi Hennig Society Meeting VII" par Swofford. Elle enrichit PAUP notamment en lui adjoignant deux types de parcimonie : Camin-Sokal et Dollo.
A ce même congrès de Stockholm, Farris a fait une démonstration de son programme Hennig86 à présent disponible. Ce programme très performant au point de vue des temps d'exécution est également capable d'éliminer les cladogrammes redondants à cause des branches de longueur nulle (cf. Platnick 88) et de manipuler des caractères non-ordonnés . Mais il ne peut traiter que la parcimonie de type Wagner (sensu stricto). Il faut encore signaler l'existence du très récent programme de compatibilité, CAFCA, écrit (en AFL) par Zandee et qui a fait l'objet d'une communication au "Willi Hennig Society Meeting VII".

Il existe déjà dans la littérature quelques études comparatives des programmes d'analyse phylogénétique pour micro-ordinateur. La plus ancienne date de 1986 (cf. Fink) et prend en compte PAUP (version 4.21), PHYLIP (version 2.8), CLINCH (version 6.2) et MacClade (version 1.0). Cette étude donne des renseignements techniques (spécifications des softwares et exigences concernant le hardware) concernant ces programmes et cite de manière non-exhaustive certaines options offertes par

chacun d'eux (possibilité de pondération,etc...). D'autre part,elle donne le résultat de tests effectués sur 3 ensembles de données avec certaines options de PAUP, PHYLIP et CLINCH au point de vue du temps d'exécution, du nombre d'arbres trouvés et de leur longueur.
D'autre part, Carpenter (1987) a réalisé un rapport succint mais particulièrement virulant à l'égard de Felsenstein d'un séminaire organisé en 1986 par ce dernier sur le thème "Computer programs for inferring phylogenies".
Les conclusions de Coddington (1987 : 179) concernant PAUP (2.4.1 et 4.4.0) et PHYLIP (2.8 et 2.9) sont plus nuancées: "With its 24 distinct programs, PHYLIP is certainly a more diverse package and it sometimes works starlingly well, but on the whole it is trickier to get quality results and more time cosumming to use."
Les deux comparaisons les plus documentées sont dues à Platnick (1987 et tableaux synthétiques distribué au "Willi Hennig Society Meeting VII"). La première étude concerne les versions 2.8 et 2.9 de PHYLIP, 2.4.0 et 2.4.1 de PAUP et le prototype SHEN qui a été incorporé dans la suite à Hennig86 . Elle teste les algorithmes exactes et ceux avec heuristiques sur pas moins de 35 ensembles de données. La seconde applique PHYLIP/MIX (version 3.1 c.à d. celle analysée dans le présent travail) et différents jeux d'options de Hennig86 (version 1.41) et de PAUP (versions 2.99 j et k) à 50 ensembles de données, cette fois. Toutes ces comparaisons s'accordent pour souligner la supériorité de PAUP sur PHYLIP tant du point de vue des temps d'exécution (efficience) que par rapport à la capacité à trouver le(s) arbre(s) le(s) plus court(s) et à détecter tous les arbres les plus parcimonieux (efficacité) .D'autre part, les tableaux synthétiques de Platnick indiquent une tendance à une efficacité accrue de Hennig86 par rapport à PAUP !
A noter enfin une rapide comparaison que vient de faire Platnick (1988) de PAUP (3.0) et de Hennig86 (édition 1.5). L'auteur y fait l'éloge de ces deux programmes gr8ce auxquels il a obtenu des résultats meilleurs que tous ceux obtenus précédemment (aussi bien sur micro-ordinateur que sur gros ordinateur).

Felsenstein répond aux critiques de Fink (1986) et de Platnick (1987) dans la documentation (cf. MAIN.DOC au paragraphe "relative speed of the different programs") accompagnant PHYLIP. Il y fait remarquer que Fink et Platnick font partie " d'une m@me école de systématiciens qui n'accorde de valeur qu'au critère de parcimonie" et qui ne fait aucun crédit aux points forts de son programme à lui à savoir : la diversité des méthodes offertes, la portabilité, le prix (PHYLIP est gratuit) et la disponibilité du code-source. Ces qualités sont réelles et méritent effectivement d'@tre soulignées ! De plus, elles sont une exclusivité de PHYLIP. Mais Felsenstein affirme d'autre part : "... MIX is undoubtably not as fast or as sophisticated as PAUP ... The quality of results from the present version of MIX may give them competition in terms of quality of result ...". Cette dernière affirmation peut @tre sérieusement mise en doute au vu des tableaux distribués par Platnick en août 88.

7.2. COMPARAISON DES ALGORITHMES HEURISTIQUES DE MIX (3.1) ET DE PAUP (2.4).

7.2.1. **Introduction.**

Ces différences d'efficacité établies, il est intéressant d'essayer de comparer les algorithmes de base utilisés dans les logiciels récents pour en dégager les points communs et les divergences. Mais malheureusement les possibilités de comparaison dans ce domaine sont particulièrement restreintes. Les algorithmes utilisés par Farris dans Hennig86 sont gardés secrets et le manuel d'utilisation de ce programme ne contient que des renseignements techniques. Par contre, même si Swofford ne communique pas ses codes-sources , il est possible de trouver de précieux renseignements d'ordre algorithmique dans le PAUP (2.4) user's manuel de cet auteur (1985) . C'est donc à PAUP que MIX sera confronté ci-dessous. La première chose à remarquer est une différence fondamentale entre ces deux programmes : le premier construit une liste d'arbres ternaires tandis que le second, une liste d'arbres binaires (aux sens définis dans le vocabulaire de la théorie des graphes). D'autre part, MIX est écrit en Pascal tandis que PAUP est rédigé en Fortran (cf. Swofford 1985 : 4-4.).
Contrairement à ce qui est affirmé p. 1-3 de user's manuel Swofford (1985), l'expérience a montré que cet auteur ne communique aucune information supplémentaire. La comparaison ci-dessous suivra donc pas à pas les indications de Swofford telles qu'il veut bien les révéler au paragraphe 3.1 de son manuel. Par convention,les options par défauts des programmes seront soulignées et les paramètres et valeurs de paramètres seront écrits en majuscules. Chaque critère est présenté d'abord de manière synthétique, puis explicité et éventuellement commenté (en mettant en parallèle les possibilités offertes respectivement par PAUP (2.4) et MIX (3.1).

7.2.2. **Comparaison des algorithmes heuristiques de base.**

7.2.2.1. Nombre d'arbre(s) conservé(s) à chaque étape d'addition séquentielle.

PAUP	MIX
.(HOLD =) 1 =	. 1
. n	. /

Explications : si HOLD = n et ADDSEQ ≠ CLOSEST (cf.7.2.2.2) alors les n meilleurs arbres sont retenus (peu importe leur longueur)

si HOLD = n et ADDSEQ = CLOSEST alors les arbres les plus parcimonieux uniquement (à concurrence de n) sont conservés .

7.2.2.2 Séquences d'addition

PAUP		MIX
(ADDSEQ =).SIMPLE (cf.Farris 1970)		. /
.ROOTLESS (cf.Farris 1970)		. /
.ASIS (ordre d'input)	=	.ordre d'input
. /		.option J=mélange aléatoire d'ordre d'input.
. CLOSEST (sélection dynamique)		. /

Explications:

Les procédures de Farris (SIMPLE et ROOTLESS) fixent à l'avance la séquence d'addition à l'aide de critères ("advancement index") basés sur les différences (phénétiques) (cf. 3.2.1.1.1.2) entre paires de taxons. Dans le cas SIMPLE, ces différences sont calculées à partir d'un point de départ fourni par l'utilisateur. Selon Farris (1970 p. 89), ce point de départ doit être l'ancêtre hypothétique commun à tous les OTUs analysés. Mais Swofford (1985 p. 3-2) suggère la possibilité de ne pas obligatoirement lier la notion d'ancêtre à celle de point de départ : il est alors possible en faisant varier ce point de modifier la séquence d'addition.
Quant à la sélection dynamique, elle choisit à chaque étape d'ajouter l'OTU qui maximise la parcimonie de l'arbre courant.

7.2.2.3 Rearrangement (ou swapping)

PAUP		MIX
(SWAP=).NO (!)		. /
.LOCAL (uniquement)		. /
.GLOBAL (uniquement)		. /
.ALT	≠	. :obligatoirement dans étape 1 : addition séquentielle-réarrangements locaux. étape 2 : réarrangements globaux (sur arbre complété).

Explications:

Dans le local swapping de Swofford des réarrangements

("nearest neighbor interchanges" (NNIs)) sont réalisés autour de chaque arête joignant des noeuds ("interior branch") selon le schéma suivant:

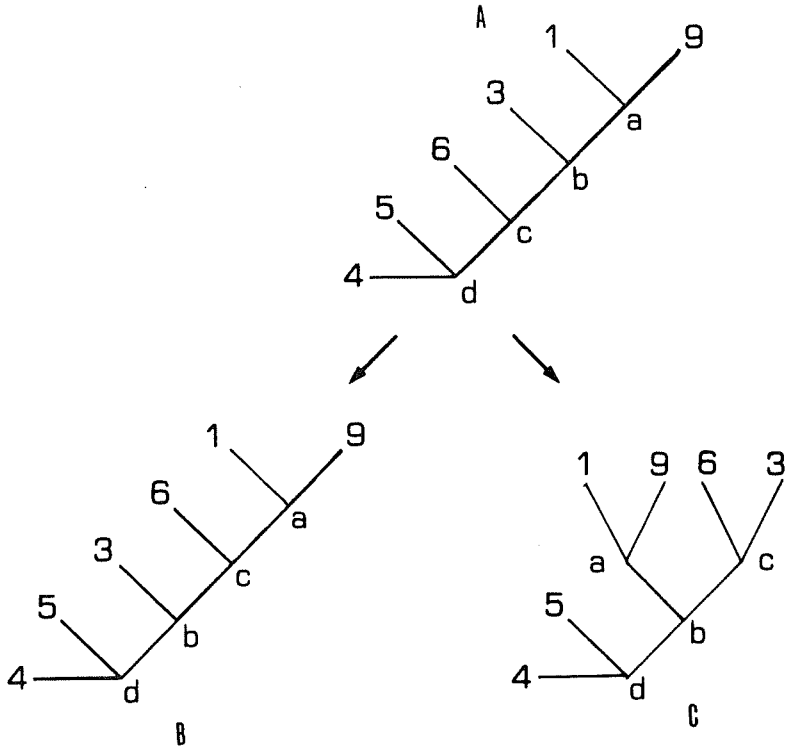


Fig. 7.1 Schéma d'un réarrangement local selon PAUP.
(adapté de user's manual 1985 p. 3.3 : des identifiants sont ici ajoutés pour les noeuds.)

Selon les termes de Swofford (1985 p. 3.3) "In global swapping , each possible subtree is removed from a tree and reinsert at all other positions on the tree".

Quant à l'option SWAP = ALT (c.-à-d. alternating) de PAUP, elle fait alterner LOCAL swapping et GLOBAL swappping de l'arbre complété jusqu'à ce qu'il n'y ait plus d'amélioration de la parcimonie.

Commentaires.

D'après les renseignements fournis par Swofford dans son manuel (1985), on peut comprendre que les réarrangements dans PAUP ne commencent que lorsque l' (/les) arbre(s) est (/sont) complété(s). (L'auteur n'est malheureusement pas très explicite à ce sujet.) Cette interprétation est confirmée par Platnick (1988). En effet cet auteur signale que dans les programmes heuristiques de Hennig86 (1.5) et PAUP (3.0) ,les réarrangements des branches ne commencent qu'après l'obtention d'une estimation initiale du meilleur arbre.

En ce qui concerne le LOCAL swapping, il faut remarquer tout d'abord que la description donnée par Swofford tout comme celle de Felsenstein dans MAIN.DOC se réfère à chaque "interior branch" comme point de départ. Dans MIX, le point de départ d'un réarrangement local est un sommet pour autant que celui-ci possède un père et un grand-père. Cette dernière clause se réfère implicitement aussi à une arête joignant deux noeuds ! On peut donc penser qu'il s'agit d'un éclairage différent d'un même processus. D'autre part,la topologie de la figure 7.1.B peut être obtenue à partir de la fig. 7.1.A si on appelle remove (3,b) puis add (c,3,b) (cf. paragraphe 6.4.2 et fig.6.5) : ceci est conforme à la réalisation d'un réarrangement local dans MIX à partir du sommet 3 puisque b est le père de 3 et c le grand-père de 3. De la même façon, la fig. 7.1.A donne la topologie de la figure 7.1.C si on appelle remove (a,b) puis add (c,a,b): ainsi donc on ne peut exclure que toutes autres choses égales, c.-à-d. abstraction faite de la différence entre arbre binaire et arbre ternaire, le processus de réarrangement local dans PAUP soit identique à celui de MIX. En revanche, comme déjà signalé précédemment,le local swapping chez Felsenstein est associé à l'addition séquentielle tandis que chez Swofford, il s'applique à un arbre complété.

Quant à la définition que Swofford donne de son GLOBAL swapping, elle n'est pas incompatible avec le processus de réarrangements globaux de Felsenstein tel que décrit ci-dessus. Elle correspond d'ailleurs presque mot pour mot à la définition de Felsenstein dans MAIN.DOC, à savoir : "Each possible subtree is removed from the tree and added back in all possible places". De plus, dans PAUP comme dans MIX, les réarrangements globaux sont réalisés sur un arbre complété.

Il faut encore ajouter que dans PAUP comme dans MIX , un processus de réarrangement sera répété jusqu'à ce qu'il n'apporte plus d'amélioration de la parcimonie.

7.2.2.4 Découverte des arbres les plus courts (s'il en existe plusieurs).

PAUP	MIX
(MULPARS) .NO	. /
.YES	. yes

Explications.

L'option MULPARS de PAUP a pour conséquence qu'est mémorisé chaque arbre ayant la même longueur que l'arbre le plus parcimonieux courant mais qui en est topologiquement distinct. MULPARS travaille en synergie avec un processus de réarrangement. Si l'utilisateur n'a pas invoqué un swapping, l'option par défaut sera LOCAL.

Commentaires.

Swofford (1985 p. 3-5) souligne que cette option MULPARS présente l'avantage supplémentaire de réduire le risque de tomber dans le piège des optima locaux. "Sometimes no single rearrangement can find a shorter tree, but a particular swap results in a tree of the same length which, if further rearranged, can then lead to a shorter tree".

7.2.2.5 Optimisation des états de variables au niveau des HTUs

PAUP offre 4 possibilités différentes d'optimisation des HTUs lors de l'addition séquentielle.(Parmi celles-ci 3 ne seront pas mentionnées ici.) L'option par défaut est OPT= FARRIS c.-à-d. la méthode proposée par cet auteur en 1970.
La première étape de cette optimisation (telle qu'elle est décrite par exemple dans Brooks ,1984 ,p. 123-128) est tout à fait analogue à un parcours postordre de l'arbre avec appel de la procédure fillin (cf. paragraphe 6.4.2). La seconde étape n' a pas de correspondant dans la partie de MIX analysée ici.

7.2.2.6 Un exemple concret.

A titre indicatif, les données de l'exemple de MIX-DOC fournies par Felsenstein pour illustrer son programme (cf. matrice de données reprise au tableau 7.1) ont été utilisées avec différentes combinaisons intéressantes d'options de PAUP décrites ci-dessus. En effet, ce programme a été implémenté sur un Olivetti M24 avec coprocesseur numérique (80 87),aux Facultés Notre-Dame de la Paix, et différentes exécutions interactives en ont été réalisées.

TABLEAU 7.1 Matrice de données fournies dans la documentation de MIX (MIX.DOC) et utilisée comme "input" de PAUP.

alpha	1	1	0	1	1	0
beta	1	1	0	0	0	0
gamma	1	0	0	1	1	0
delta	0	0	1	0	0	1
epsil	0	0	1	1	1	0

Les résultats obtenus sont consignés dans le tableau 7.2 et deux listings correspondants sont repris à titre exemplatif dans les fig. 7.2 et 7.3 . Ils permettent de confirmer la supériorité de l'efficacité de PAUP par rapport à MIX.

Commentaires.

Swofford (1985 p. 3-5) souligne que cette option MULPARS présente l'avantage supplémentaire de réduire le risque de tomber dans le piège des optima locaux. "Sometimes no single rearrangement can find a shorter tree, but a particular swap results in a tree of the same length which, if further rearranged, can then lead to a shorter tree".

7.2.2.5 Optimisation des états de variables au niveau des HTUs

PAUP offre 4 possibilités différentes d'optimisation des HTUs lors de l'addition séquentielle.(Parmi celles-ci 3 ne seront pas mentionnées ici.) L'option par défaut est OPT= FARRIS c.-à-d. la méthode proposée par cet auteur en 1970.
La première étape de cette optimisation (telle qu'elle est décrite par exemple dans Brooks ,1984 ,p. 123-128) est tout à fait analogue à un parcours postordre de l'arbre avec appel de la procédure fillin (cf. paragraphe 6.4.2). La seconde étape n' a pas de correspondant dans la partie de MIX analysée ici.

7.2.2.6 Un exemple concret.

A titre indicatif, les données de l'exemple de MIX-DOC fournies par Felsenstein pour illustrer son programme (cf. matrice de données reprise au tableau 7.1) ont été utilisées avec différentes combinaisons intéressantes d'options de PAUP décrites ci-dessus. En effet, ce programme a été **implanté** sur un Olivetti M24 avec coprocesseur numérique (80 87),aux Facultés Notre-Dame de la Paix, et différentes exécutions interactives en ont été réalisées.

TABLEAU 7.1 Matrice de données fournies dans la documentation de MIX (MIX.DOC) et utilisée comme "input" de PAUP.

alpha	1	1	0	1	1	0
beta	1	1	0	0	0	0
gamma	1	0	0	1	1	0
delta	0	0	1	0	0	1
epsil	0	0	1	1	1	0

Les résultats obtenus sont consignés dans le tableau 7.2 et deux listings correspondants sont repris à titre exemplatif dans les fig. 7.2 et 7.3 . Ils permettent de confirmer la supériorité de l'efficacité de PAUP par rapport à MIX.

TABLEAU 7.2 Résultats obtenus avec différentes combinaisons d'options de PAUP pour les données de l'exemple fournies par Felsenstein dans MIX.DOC

1.ADDSEQ	CLOSEST	SIMPLE	ROOTLES	ASIS	ASIS	ASIS	ASIS	ASIS	ASIS
2.HOLD	1	1	1	1	2	1	1	1	1
3.SWAP	NO	NO	NO	NO	NO	ALT	GLOBAL	LOCAL	LOCAL
4.MUL-PARS	NO	NO	NO	NO	NO	NO	NO	NO	YES
Longueur de l'arbre	8	8	8	9	8	8	8	9	8 !

Légende : Les options par défaut sont soulignées.
L'optimisation est toujous celle de Farris.

En effet, alors que ce dernier programme obtient un arbre minimal dont la longueur est 9, presque toutes les combinaisons testées de PAUP font mieux ! Pour ces données, les séquences d'additions CLOSEST, SIMPLE et ROOTLES offertes par PAUP sont plus performantes même sans aucun swapping (mais avec une optimisation !) que l'algorithme avec double swapping de Felsenstein ! D'autre part, si on sélectionne les options ADDSEQ=ASIS et HOLD=1 (ce qui rejoint l'algorithme de Felsenstein dans l'exemple de MIX-DOC),on peut constater que les réarrangements globaux (GLOBAL) ou alternés (ALT) seuls sont plus efficaces que ceux de MIX en ce qui concerne les données en question.Et enfin, on a ici la confirmation du commentaire de Swofford à propos de l'option MULPARS : elle peut permettre d'améliorer la parcimonie de l'arbre.

Il est regrettable que, sur les données choisies par Felsenstein comme exemple (cf. tableau 7.1), les options par défaut de MIX ne fournissent pas l'arbre de Wagner (au sens strict) dont la parcimonie est maximale (cf. MIX.DOC)... mais il faut remarquer qu'avec l'option J de MIX , j'ai pu obtenir un arbre de longueur 8 (voir fig. 5.2) .

```
Option settings:

     NOTU ..................          5
     NCHAR .................          6
     User-tree(s) ..........         NO
     HYPANC ................          1
  ►  ADDSEQ ................     SIMPLE
  ►  HOLD ..................          1
  ►  SWAP ..................         NO
  ►  MULPARS ...............         NO
     OPT ...................    FARRIS
     ROOT ..................  ANCESTOR
     Weights applied .......         NO
     OUTWIDTH ..............         80
     Missing data code .....       None
     MAXTREE ...............        N/A


Branch lengths and linkages for unrooted tree no.   1

                          Connected          Branch
      Node                to node            length

     beta (   2)              8               2.000
    gamma (   3)              7               0.000
    delta (   4)              6               3.000
    epsil (   5)              6               0.000
        6                     7               2.000
        7                     8               1.000
        8                   alp (   1)        0.000


Statistics for tree no.   1

  ►     Length =       8.000
        Consistency index = 0.750


Tree no.   1 rooted using designated ancestor


* alp    1
*
********************** beta    2
8
*           * gamma    3
***********7
            *                          ********************************** delta    4
            **********************6
                                     * epsil    5
```

Fig. 7.2 "Output" de PAUP obtenant un arbre plus court que dans l'exemple de MIX.DOC.

```
Option settings:

     NOTU .....................        5
     NCHAR ....................        6
     User-tree(s) .............       NO
     HYPANC ...................        1
   ► ADDSEQ ...................     ASIS
   ► HOLD .....................        1
   ► SWAP .....................    LOCAL
   ► MULPARS ..................       NO
     OPT ......................   FARRIS
     ROOT ..................... ANCESTOR
     Weights applied .........       NO
     OUTWIDTH .................       80
     Missing data code .......     None
     MAXTREE ..................      N/A



Branch lengths and linkages for unrooted tree no.  1


                          Connected          Branch
      Node                to node            length

     beta (  2)              6                1.000
    gamma (  3)              7                0.000
    delta (  4)              6                3.000
    epsil (  5)              7                2.000
        6                    8                2.000
        7                    8                0.000
        8                  alp (  1)          1.000


Statistics for tree no.  1

   ►    Length =      9.000
        Consistency index = 0.667


Tree no.  1 rooted using designated ancestor


* alp   1
*
*                                  ********** beta   2
*           ********************6
*           *                      ********************************** delta   4
***********8
            * gamma   3
            7
            ********************** epsil   5



* alp   1
*
*                                          ********************* beta   2
*                       ********************6
*                       *                  ********************* delta   4
**********************8
*                       *                  ********************* gamma   3
*                       ********************7
                                           ********************* epsil   5
```

Fig. 7.3 "Output" de PAUP obtenant un arbre de même longueur que dans l'exemple de MIX.DOC .

7.2.3. **Notes relatives à l'enracinement cladistique et aux types d'arbres.**

Alors que Felsenstein a choisi d' implémenter un arbre binaire (avec 2spp-1 sommets) dans MIX, Swofford, lui ,construit un ou plusieurs arbres ternaires (avec [2 x nombres d'OTUs - 2] sommets).

D'autre part dans PAUP, les arbres ne sont jamais construits d'emblée enracinés cladistiquement. La situation est donc plus complexe dans MIX puisque, selon les cas de figure, l'évaluation de la parcimonie se fera sur un arbre enraciné cladistiquemnet ou non. On peut se demander, à ce propos, si les résultats pouvant être obtenus avec un enracinement cladistique primaire d'un arbre ternaire sont strictement équivalents à ceux pouvant être obtenus à partir d'un arbre ternaire construit non-enraciné cladistiquement et postérieurement enraciné. Cette question s'inscrit dans le prolongement de la remarque suivante faite par Farris (1970 , p.89) lorsqu'il compare son algorithme construisant un arbre d'emblée enraciné ("SIMPLE WAGNER METHOD") et celui produisant un arbre d'abord non-enraciné ("ROOTLESS WAGNER METHODS ") : "The simple Wagner algorithm does not impose irreversibility on the trees it produces, and from this standpoint, the choice of an ancestor may not be crucial. It has been found, however, that the form of the tree produced by a simple Wagner algorithm can be changed by altering the ancestor ...".

Enfin, à partir d'un arbre ternaire non-enraciné cladistiquement, Swofford offre 4 modes possibles d'enracinement. Parmi ceux-ci les plus intéressants sont ROOT=ANCESTOR (option par défaut) et ROOT=OUTGROUP. Cette dernière option est tout à fait analogue à l'option O (c.-à-d. enracinement cladistique secondaire) de MIX. Par contre, la première n'a pas son correspondant exact dans le programme de Felsenstein, puisque comme cela vient d'être souligné, l'enracinement par ancêtre se fait d'emblée (enracinement cladistique primaire) dans MIX. De plus, selon la grille d'analyse définie au paragraphe 6.5.3., la racine obtenue lorsque ROOT=ANCESTOR dans PAUP sera toujours de degré 1 (et l'arbre reste donc ternaire). En revanche, elle peut être de degré 1 (arbre ternaire) ou 2 (arbre binaire) lors d'un enracinement cladistique primaire dans MIX. Enfin, les conventions imposées par les 2 auteurs sont légèrement différentes : chez Swofford, l'ancêtre commun est à déclarer parmi les OTUs tandis que chez Felsenstein,il ne peut pas être inclus dans le spp OTUs à analyser et est défini à part.

CONCLUSIONS.

Ce mémoire a permis d'amorcer la description d'une utilisation non-conventionnelle de l'informatique : celle relative à l'inférence de cladogrammes.

Il a réalisé tout d'abord une revue synthétique de la littérature cladistique, soulignant à plusieurs reprises la participation croissante prise par les programmes dans l'analyse des caractères (pour éviter le recours à des hypothèses à prio - ri) , s'intéressant au codage de ces derniers ,privilégiant les approches algorithmiques et se focalisant sur les études de logiciels disponibles.

Cette étude a proposé une comparaison de l'analyse phylogénétique avec les langages formels dans le but de cerner avec davantage de rigueur et de profondeur le problème posé. Car une bonne définition de celui-ci est bien évidemment une condition nécessaire à la réalisation d' un algorithme programmable pour le résoudre.
De plus cette nouvelle approche pourrait peut-être ,dans l'avenir , permettre un enrichissement de l'analyse cladistique elle-même. En effet, la théorie des grammaires formelles est actuellement très développée et il n'est pas impossible que certains de ses théorèmes soient adaptables à l'inférence d'arbres phylogénétiques.

Enfin, l'explicitation formelle du concept d'optimisation (impliqué par cette inférence) qui a été réalisée dans ce travail pourrait ouvrir la voie à la possibilité de découverte d'autres concepts de parcimonie.

L'informatisation de l'analyse cladistique est actuellement en plein essor. Mais , paradoxalement, très peu de publications décrivent les algorithmes utilisés. En réponse à cette carence, ce mémoire a fourni des spécifications (au sens informatique) à une solution proposée par Felsenstein . Il décrit en détail deux sujets qui ne sont que vaguement évoqués dans la littérature à savoir une heuristique appelée "branch swapping" ou réarrangement et l'utilisation par le programme de la valeur inconnue.

La solution offerte par MIX à la recherche des topologies les plus parcimonieuses a été étudiée en détail . En revanche, l'optimisation des valeurs des variables au niveau des HTUs ,pour une topologie donnée n'a été qu'effleurée et ce sujet mériterait d'être approffondi.
Un seul des nombreux logiciels de PHYLIP a été pris en compte ici. Les codes-sources d'autres programmes d'analyse cladistique réalisés par Felsenstein , avec heuristique (METRO par exemple) ou sans (PENNY , DOLLOP ...) , pourraient utilement être scrutés et comparés à ceux de MIX.
D'autre part, une description et une mise en parallèle des algorithmes de PAUP (3.0) et de Hennig86 serait extrêmement intéressante.

TABLE DES FIGURES.

TABLE DES TABLEAUX.

INDEX

Cet index ne reprend que les termes fréquemment utilisés dans ce mémoire ,avec les pages correspondantes. Il a pour but essentiel d'aider le lecteur à retrouver facilement les définitions de ces mots.

B I B L I O G R A P H I E

ANDREWS, P. & L. MARTIN.
1987 Cladistic relationships of extant and fossil hominoids.
J. Hum. Evol., 16, 101-118.

[ARCHIE, J.W.
1985 Methods for coding variable morphological features for numerical taxonomic analysis.
Syst. Zool., 34 : 326-345.]
cité dans Goldman, 1988.

ARCHIE, J.W.
1987 Summary of the Twentieth International Numerical Taxonomy Conference.
Syst. Zool., 36(2) : 216-223.

ARNOLD, E.N.
1981 Estimating phylogenies at low taxonomic levels.
Z. Zool.Syst.Evolut.-Forsch. 19, 1-35.

* BARTHELEMY, J.P. & A. GUENOCHE.
1988 Les arbres et les représentations des proximités.
Paris, Milan, Barcelone, Mexico, Masson, 240 p.

[BEATTY, J.
1982 Classes and cladists.
Syst. Zool. ,31 : 25-34.]
cité dans Kluge, 1985.

* BENZECRI, J.P. & Collaborateurs.
1984 L'analyse des données.
1 La taxinomie.
Paris, Dunod ,635 p.

BLACKMORE, S.B.
1986 Cellular ontogeny.
Cladistics 2(4) : 358-362.

[BOYDEN, A.
1947 Homology and analogy. A critical review of the meanings and implications of these concepts in biology.
Amer. Midl. Nat. : 37 : 648-669.]
cité dans Hennig 1966.

[BREMER, K. & H.-E. WANNTORP,
1979 Geographic populations or biological species in phylogeny reconstruction?
Syst. Zool., 28 : 220-224.]
cité dans de Queiroz et Donoghue ,1988.

BROOKS, D.R.
1984 Quantitative parsinomy.
in Duncan,Th.& T.F. Stuessy ed ,Cladistics Perspectives on the Reconstruction of Evolutionary History.
New York, Columbia University Press :119-132.

BROOKS, D.R. & E.O. WILEY
1985 Theories and methods in different approaches to phylogenetic systematics.
Cladistics 1(1) : 1-11.

[BROOKS, D.R. & E.O. WILEY
1986 Evolution as entropy : toward a unified theory of biology.
Chicago, the University of Chicago Press, 335 p.]
cité dans Farris ,1989.

CAMIN, J.H. & J.R. SOKAL
1965 A Method for deducing branching sequences in phylogeny.
Evolution 19 : 311-326.

* CARPENTER, J.M.
1987 A report on the society for the study of evolution workshop "Computer programs for inferring phylogenies".
Cladistics 3(1) : 52-55.

[CAVALLI- SFORZA, L.L. A.W.F. EDWARDS
1967 Phylogenetic analysis : models and estimation procedures.
Evolution 21 : 550-570.]
cité dans Mickevich & Platnick ,1989.

CHALINE, J.
1983 Le gradualisme phylétique et la théorie synthétique de l'évolution.
Archeologia 73 : 46-51.

CHALINE, J.
1983a Les processus de spéciation.
Archeologia 73 : 30-32.

CLARK, C. & D.J. CURRAN
1986 Outgroup analysis, homoplasy, and global parsimony: a response to Maddison, Donoghue, and Maddison.
Syst. Zool. 35(3) : 422-426.

CODDINGTON, J.A.
1986 Orb Webs in "Non-orb weaving" ogre-faced spiders (Araneae : Dinopidae) : A question of genealogy.
Cladistics 2(1): 53-67.

CODDINGTON, J.A.
1987 The sixth annual meeting of the Willi Hennig Society.
Cladistics 3(2) : 178-185.

COLLESS, D.H.
1985 On the status of outgroups in Phylogenetics.
Syst. Zool. 34(3):364-366.

* E COOK
1971 The complexity of problem solving procedures.
Proc. Third Ann. A.C.M. Symp.on Theory of Computing, Ass. for Comp. machinery , New York : 151-158.]
Cité dans BARTHELEMY & GUENOCHE 1988.

CRACRAFT, J.
1981 The use of functional and adaptive criteria in phylogenetic systematics.
Amer. Zool., 21:21-36.

CRANSTON, P.S. & C.J. HUMPHRIES
1988 Cladistics and computers : a Chironomid conundrum ?
Cladistics 4(1) : 72-92.

E CUENOT, L.
1940 Remarques sur un essai d'arbre généalogique du règne animal.
Compte Rendu Hebdomadaire des Séances de l'Académie des Sciences ,210 : 23-27.]
Cité dans Tassy 1986.

[DARWIN, C.
1859 On the origin of species , a facsimile of the first edition (1859) with introduction by Ernest Mayr (1966). Cambridge ,Mass. ; Harvard Univerity Press.] cité dans Hull 1979.

* DAY, W.H.E.
1983 Computationally difficult parsimony problems in phylogenetic systematics. J. Theor. Biol. 103 : 429-438.

* DAY, W.H.E., D.S. JOHNSON & D. SANKOFF.
1986 The computanional complexity of inferring rooted phylogenies by parsimony. Math. Biosc. 81 : 33-42.

* DAY, W.H.E. & D. SANKOFF.
1986 Computational complexity of inferring phylogenies by compatibility. Syst. Zool., 35(2) : 224-229.

* DAY, W.H.E. & D. SANKOFF.
1987 Computational complexity of inferring phylogenies from chromosome inversion data. J. Theor. Biol. 124, 213-218.

DEBRY, R.W. & N.A. SLADE.
1985 Cladistic analysis of restriction endonuclease cleavage maps within a maximum-likelihood framework. Syst. Zool., 34(1):21-34.

DELSON, E.
1976 Catarrhine phylogeny and classification : principles , methods and comments. J. Hum. Evol. 6 : 433-459.

DELSON, E. , N. ELDREDGE & I. TATTERSALL
1977 Reconstruction of Hominid Phylogegy : A Testable Framework Based on Cladistic Analysis. J. Human Evol. 6 : 263-278.

* DIEUDONNE, J.
1987 Pour l'honneur de l'esprit humain. Les mathématiques aujourd'hui. Hachette, 298 p.

E DOLLO, L.
1893 Les lois de l'évolution. Bull. Soc. belge Géol. Pal. Hydr. 7 :164-166.] cité dans Hennig 1966.

E DUNCAN, T.
1984 Willi Hennig , character compatibility, and the "dendogrammaceae" revisited. Taxon,33 : 698-704.] cité dans Farris , 1985a

E EDWARDS, A.W.F. & L. CAVALLI-SFORZA
1963 The reconstruction of evolution. Ann. Hum. Genet. 27 : 105] cité dans Felsenstein 1983.

E ELDREDGE, N. & S.J. GOULD
1972 Punctuated equilibria : an alternative to phyletic gradualism. In : Schopf , T.J. eds., Models in paleobiology San Francisco, Freeman, Cooper & Co. : 82-155.] cité dans Mindell et al., 1989.

ELDREDGE, N. & I. TATTERSALL,
1975 Evolutionary models, phylogenetic reconstruction, and another look at hominid phylogeny. In Szalay ,Approaches to Primates Paleobiology Contr. Primat. ,Basel , Karger , 5 :218-242.

ENGELMANN, G.F. & E.O. WILEY.
1977 The place of ancestor-descendant relationships in phylogeny reconstruction. Syst. Zool. 26: 1-11.

E ESTABROOK, G.F.
1968 A general solution in partial orders for the Camin-sokal model in phylogeny. J. Theor. Biol. 21 : 421-438.] cité dans Farris 1970.

* FARRIS, J.S.
1970 Methods for computing Wagner trees.
Syst. Zool. 19: 83-92.

FARRIS, J.S.
1971 The hypothesis of nonspecificity and taxonomic congruence.
Ann. Rev. Ecol. Syst., 2 : 277-302.

* FARRIS, J.S.
1972 Estimating phylogenetic trees from distance matrices.
American Naturalist, vol.106,n°951 :645-668.

FARRIS, J.S.
1977a Phylogenetic analysis under Dollo's law.
Syst. Zool. 26: 78-88.

FARRIS, J.S.
1977b Some Further Comments on Le Quesne's Methods.
Syst. Zool.: 220-223.

[FARRIS, J.S.
1978 Inferring phylogenetic trees from chromosome inversion data.
Syst. Zool. 27 : 275-284.]
cité dans Day & Sankoff, 1987.

FARRIS, J.S.
1979 The information content of the phylogenetic system.
Syst. Zool. 28 : 483-519.

[FARRIS, J.S.
1980 a Naturalness, invariance , and the consequences of phenetic criteria.
Syst. Zool. 29 : 360-381.]
cité dans Farris 1982 b.

FARRIS, J.S.
1980 b The efficient diagnoses of the phylogenetic system.
Syst. Zool. 29 : 386-401.

FARRIS, J.S.
1982 a Outgroups and parsimony.
Systematic Zoology 31(3) : 328-334.

FARRIS, J.S.
1982 b Simplicity and informativeness in systematics and phylogeny.
Syst. zool. 31(4) : 413-444.

[FARRIS, J.S.
1983 The logical basis of phylogenetic analysis.
in Funk,V,A & N.I. Platnick eds.
Advances in cladistics II
Proc. Willy Hennig Soc.
New York, Columbia Univ. Press, 2 :7 -36.]
cité dans Farris 1982 b.

FARRIS, J.S.
1985 Reviews
Cladistics : perspectives on the reconstruction of evolutionary history.
Cladistics 1(3) : 292-299.

FARRIS, J.S.
1985 a Parsimony, synapomorphy ,and explanatory power: a reply to Duncan.
Taxon,34 : 130-135.

* FARRIS, J.S.
1986a Distances and Statistics.
Cladistics 2(2): 144-157.

FARRIS, J.S.
1986b On the boundaries of phylogenetic systematics.
Cladistics 2(1) : 14-27.

FARRIS, J.S.
1989 Entropy and fruit flies.
Cladistics 5 : 103-108.

FARRIS, J.S. & A.G. KLUGE
1986 Synapomorphy, parsimony and evidence.
Taxon, 35 : 298-306.

* FARRIS, J.S., A.G. KLUGE & M.J. ECKARDT.
1970 A numerical approach to phylogenetic systematics.
Syst. Zool. 19: 172-191.

FELSENSTEIN, J.
1978 Cases in wich parsimony and compatibility methods will be positively misleading.
Syst. Zool. 27 : 401-410.

* FELSENSTEIN, J.
1979 Alternative methods of phylogenetic inference and their interrelationship.
Syst. Zool., 28: 49-62.

* FELSENSTEIN, J.
1982 Numerical methods for inferring evolutionary trees.
Q. Rev. Biol., 57, (4):379-404.

* FELSENSTEIN, J.
1983 Parsimony in Systematics : Biological and Statistical Issues.
Ann. Rev. Ecol. Syst. 14:313-333.

* FELSENSTEIN, J.
1986 Distance Methods : A reply to Farris.
Cladistics 2(2): 130-143.

* FICHEFET, J.
Théories des graphes et son algorithmique.
Notes de cours.

* FINK, W.L.
1986 Microcomputers and phylogenetic analysis.
Science 234 : 1135-1139.

*[FITCH, W.M.
1971 Toward defining the course of evolution: minimum change for a specific tree topology.
Syst. Zool. : 20 : 406-416.]
cité dans Swofford, 1985.

* FORSTER, M.R.
1986 Statistical covariance as a measure of phylo-genetic relationship.
Cladistics 2 (4) : 297-317.

* FOULDS, L.R. & R. L. GRAHAM.
1982 The steiner problem in phylogeny is NP-Complete. Advances in applied mathematics 3, 43-49.

* FRIJTERS, D. & A.,LINDENMAYER
1974 A model for the growth and flowering of Aster Novae-Angliae on the basis of table (1,0) L-Systems
in Goos,G. & J.,Hartmanis, eds.
Lecture Notes in Computer Science, Berlin, Heidelberg, New York; Springer Verlag : 24-52.

GAFFNEY, E.S.
1979 An introduction to the logic of phylogeny reconstruction.
in Cracraft ,J. & N. Eldredge,
Phylogenetic analysis and paleontology.:79-111.
Ed.J. Cracraft and N. Eldredge.;New York,
Columbia University Press.

GAUTHIER, J., A.G. KLUGE & T. ROWE.
1988 Amniote phylogeny and the importance of fossils.
Cladistics 4: 105.

* GOLDMAN, N.
1988 Methods for discrete coding of morphological characters for numerical analysis.
Cladistics 4(1) : 59-71.

* GOLDMAN, N.
1989 Fewest variables coding method for multistate characters.
Syst. Zool.38(1) : 79-85.

* GOLDSCHLAGER, L. & A. LISTER
1986 Informatique et algorithmique.
Intereditions, Prentice Hall International, 310 p.

E GOULD, S.J.
1982 The meaning of punctuated equilibrium and its role in validating a hierarchical approach to macroevolution.
In : Milkman, R. ,ed. Perspectives in evolution
Sunderland, Massachusetts, Sinauer. : 83-104.]
cité dans Mindell et al. 1989.

* GRAHAM, R.L. & L.R. FOULDS.
1982 Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time.
Math. Biosc. 60: 133-142.

* GOGUEN, J.A., J.W. THATCHER, E.G. WAGNER & J.B. WRIGHT.
1973 A junction between computer science and category theory I.
Mathematical Sciences Department. IBM Thomas J. Watson research center. Yorktown Heights, New York 10598: 108 p.

HENNIG, W.
1966 Phylogenetic Systematics.
University of Illinois Press, Urbana, Chicago, London. 263 p.

E HENNIG, W.
1983 Stammesgeschichte der Chordates.
Verlag Paul Farey, Hamburg.]
cité dans Farris, 1985 a.

* E HENDY, M.D. & D. PENNY
1982 Branch and bound algorithms to determine minimal evolutionary trees.
Math. Biosc. 59 : 277-290.]
cite dans Swofford 1985.

* HOPCROFT, J.E. & J.D. ULLMAN.
1969 Formal languages and their relation to automata.
Reading, Massachusetts, Menlo Park, California. London, Amsterdam, Don Mills, Ontario, Sydney. Addison-Wesley Publishing Company. 242 p.

HULL, D.L.
1979 The limits of cladism.
Syst. Zool. 28 : 416-440.

HUMPHRIES, C.J. & J.M. CAMUS
1986 Contemporary issues in systematics.
Cladistics 2(1) : 85-99.

E HUXLEY, J.S.
1967 The three types of evolutionary process.
Nature 180 : 454-455.]
Cité dans Tassy ,1986.

JANVIER, Ph., P. TASSY, & H. THOMAS
1980 Le Cladisme.
La recherche,11 (117) :1396-1406.

KLUGE, A.G.
1985 Ontogeny and phylogenetic systematics
Cladistics 11 : 13-27.

KLUGE, A.G.
1989 A concern for evidence and a phylogenetic hypothesis of relationships among Epicrates (Boidae, Serpentes).
Syst. Zool. 38(1) :7-25.

KLUGE, A.G. & J.S. FARRIS
1969 Quantitative Phyletics and the Evolution of Anurans.
Syst. Zool. 18 : 1-32.

KOCIOLEK, J.P. & D.M. WILLIAMS
1987 Unicell ontogeny and phylogeny:
examples from the diatoms.
Cladistics 33) : 274-284.

KRAUS, F.
1988 An empirical evaluation of the use of the ontogeny polarization criterion in phylogenetic inference.
Syst. Zool. 37(2) : 106-141.

* [LINDENMAYER, A.
1968 a Mathematical models for cellular interactions in development. Part 1
J. Theor. Biol. 18 :280-290.]
cité dans Rozenberg 1974.

* [LINDENMAYER, A.
1968 b Mathematical models for cellular interactions in development. Part 2.
Jour. Theor. Biol. 18 :300-315.]
cité dans Rozenberg 1974.

LE QUESNE, W.J.
1975 The uniquely evolved character concept and its cladistic application.
Syst. Zool. 23: 513-517.

LE QUESNE, W.J.
1977 The uniquely evolved character concept.
Syst. Zool. :218-220.

LE QUESNE, W.J.
1982 Compatibility analysis and its applications.
Zoological journal of the linnean society,
74: 267-275.

* LEWIS, H.R. & C.H. PAPADIMITRIOU
1978 L'efficacité des algorithmes.
Pour la Science : 62-75.

LØVTRUP, S.
1986 Evolution, morphogenesis, and recapitulation:
an essay on metazoan evolution.
Cladistics: 2(1) :68 -82.

* LUCKOW, M. & R.A. PIMENTEL.
1985 An empirical comparison of numerical Wagner
computer programs.
Cladistics 1(1): 47-66.

E LUNDBERG, J.G.
1972 Wagner networks and ancestors.
Syst. Zool. , 21 : 398-413 .]
Cité dans Swofford, 1985.

MADDISON, W.P., M.J.DONOGHUE & D.R. MADDISON
1984 Outgroup analysis and parsimony.
Syst. Zool. 33(1): 83-103.

* MANDRIOLI, D. & C. GHEZZI
1987 Theoretical foundations of computer science.
New York, Chichester, Brisbane, Toronto, Singapore
Wiley & Sons , 478 p.

MASLIN, T.P.
1952 Morphological criteria of phyletic relation-
ships.
Syst. Zool. ,1 : 49-70.

E MAYR, E.
1969 Principles of numerical taxonomy.
Mc Graw Hill, New York.]
cité dans Tassy 1986.

[MAYR , E.
1988 Toward a new Philosophy of biology
Cambridge ,Massachusets, Harvard University Press]
cité dans Mindell et al. 1989.

MEACHAM, C.A.
1981 A manual method for character compatibility ana-
lysis.
Taxon, 30(3) : 591-600.

MEACHAM, C.A.
1984 The role of hypothesized direction of characters
in the estimation of evolutionary history.
Taxon 33(1): 26-38.

* MICKEVICH, M.F.
1982 Tranformation Series Analysis.
Syst. Zool. 31(4) : 461-478.

MICKEVICH, M.F. & N.I. PLATNICK
1989 On the information content of classifications.
Cladistics 5 : 33-47.

[MILES, R.S.
1973 Relationships of acanthodians.
in Greenwood P.H. ,R.S. Miles & C. Patterson, eds
Interrelationships of fishes, Londres,Academic
Press : 63-103.]
cité dans Tassy, 1986.

MINDELL, D.P. , J.W. SITES, Jr. & D. GRAUR
1989 Speciational evolution : a phylogenetic test with
allozymes in Sceloporus (Reptilia).
Cladistics 5 : 49-61.

[NAEF, A.
1931 Phylogeny der Tiere.
in Baur & Hartmann ,eds ,Handbuch der Vererbungs-
wissenschaft III, 1 (Liefg. 13).]
cité dans Hennig, 1966.

NELSON, G.
1978 Ontogeny, phylogeny, paleontology, and the
Biogenetic Law.
Syst. Biol. 27 :324-345.

NELSON, G.
1985 Outgroups and ontogeny.
Cladistics 1(1) : 29-45.

[NELSON, G. & N.I. PLATNICK
1981 Systematics and biogeography
Cladistics and Vicariance.
Columbia University Press ,New York, 567 p.]
cité par Kociolek & Williams , 1987.

* O'GRADY, R.T. & G.B. DEETS
1987 Coding multistate characters ,with special reference to the use of parasites as characters their hosts
Syst. Zool. ,36(3): 268-279.

* O' GRADY, R.T. , G.B. DEETS & G.W. BENZ
1988 Additional observations on non-redundant linear coding of multistate characters.
Syst. Zool. 38(1) : 54-57.

[PATTERSON, C.
1982 Morphological characters and homology.
in Joysey, K.A. & A.E. Friday, eds.
Problems in phylogenetic reconstruction
London, Academic Press , 21 : 21-74.]
cité dans Kraus , 1988.

PIMENTEL, R.A. & R.,RIGGINS
1987 The nature of cladistic data.
Cladistics 3(3) :201-209.

PLATNICK, N.I.
1985 Philosophy and the transformation of cladistics revisited.
Cladistics 1(1) : 87-94.

PLATNICK, N.I.
1986 On justifying cladistics.
Cladistics 2(1) : 83-85.

* PLATNICK, N.I.
1987 An empirical comparison of microcomputer parsimony programs.
Cladistics 3(2): 121-144.

* PLATNICK, N.I.
1988 Programs for quicker relationships.
Nature 335 : 310.

* [POLYA, G.
1957 How to solve it : a new aspect of the mathematical method.
Princeton,NJ; Princeton Univ. Press.]
cité dans Veloso ,1982.

* [POLYA, G.
1962 Mathematical discovery : on understanding, learning and teaching problem solving
New York, Wiley.]
cité dans Veloso ,1982.

[POPPER, K.R.
1968 The logic of scientific discovery.
Harper Torchbooks , New York.]
cité dans Engelmann & Wiley, 1977.

de QUEIROZ, K.
1985 The ontogenetic method for determining character polarity and its relevance to phylogenetic systematics.
Syst. Zool., 34(3): 280-299.

de QUEIROZ, K. & M.J. DONOGHUE,
1988 Phylogenetic Systematics and the Species Problem.
Cladistics 4(4) : 317-338.

* ROHLF, F.J.
1984 A note on minimum length trees.
Syst. Zool., 33(3) : 341-343.

[ROSEN, D.E.
1978 Vicariant patterns and historical explanations in biogeography.
Syst. Zool. ,27 : 159 -188.]
cité dans de Queiroz et Donoghue ,1988.

[ROSEN, D.E.
1979 Fishes from the uplands and intermontane basins of Guatemala : revisionary studies and comparative geography.
Bull. Amer. Mus. Nat. Hist. , 162 : 267-376.]
cité dans de Queiroz et Donoghue, 1988.

[ROSEN, D.E.
1982 Do current theories of evolution satisfy the basic requirements of explanations.
Syst. Zool., 31 : 76-85.]
cité dans Kraus, 1988.

* ROZENBERG, G.
1974 Theory of L Systems :
From the point of view of Formal Language Theory.
in Goos G. & J. Hartmanis eds.,
Lecture Notes in Computer Science,
Berlin, Heidelberg,New York; Springer Verlag :1-23

SÆTHER, O.A.
1986 The Myth of objectivity - Post-Hennigian deviations.
Cladistics 2 : 1-13.

[SIMPSON, G.G.
1961 Principles of animal taxonomy.
Columbia University Press, New York]
cité dans Janvier et al. 1980.

SKELTON, R.R., H.M. McHENRY & G.M. DRAWHORN
1986 Phylogenetic analysis of early hominids.
Current Anthroplogy, 27 (1) :21 -43.

SOBER, E.
1985 A likelihood justification of parsimony.
Cladistics 1(3): 209-233.

[SOKAL,R.R. & P.H.A. SNEATH,
1963 Principles of numerical taxonomy.
W.H. Freeman, San Francisco.]
cité dans Tassy 1986.

STEBBINS, G.L. and F.J. AYALA
1985 The evolution of Darwinism.
Sc. Amer. 253(1): 54-64.

STEVENS,P.F.
1980 Evolutionary polarity of character states.
Ann. Rev. Ecol. Syst., 11: 333-358.

* SWOFFORD, D.L.
1985 PAUP
Phylogenetic Analysis Using Parsimony
Version 2.4 User's manual.
Illinois Natural History Survey
607 East Peabody Drive, Champaign, Illinois 61820

* SWOFFORD, D.L. & S.H. BERLOCHER.
1987 Inferring evolutionary trees from gene frequency data under the principle of maximum parsimony.
Syst. Zool., 36(3): 293-325.

SZALAY, F.S.
1981 Functional analysis and the practice of the phylogenetic method as reflected by some mammalian studies.
Amer.Zool. 21 : 37-45.

TASSY, P.
1986 Construction systématique et soumission au test : une forme de connaissance objective.
dans L'ordre et la diversité du vivant.
Quel statut scientifique pour les classifications biologiques ?
Fondation Diderot, Librairie Arthème Fayard. : 84-98 et 253-269.

TASSY, P.
1988 The classification of proboscidea : how many cladistic classifications ?
Cladistics 4: 43-57.

TASSY, P. & P. DARLU
1988 Les analyses phylogénétiques informatisées: Une révolution en anthropologie?
Bul. Mém. Soc. Anthr. Paris 4(4) : 293-296.

TAYLOR, J.T.
1987 Historical versus selectionist explanations in evolutionary biology.
Cladistics 3(1) : 1-13.

THOMPSON, E.A.
1986 Likelihood and parsimony : comparison of criteria and solutions.
Cladistics 2(1) : 43-52.

THUILLIER, P.
1983 Enquète: le "scandale" du British Museum.
La Recherche 125 : 1016-1023.

TURNER, A. & A. CHAMBERLAIN
1989 Speciation, morphological change and the status of African Homo erectus.
J. Hum. Evol. 18 ,115-130.

* VELOSO, P.A.S.
1982 Outlines of a mathematical theory of problems.
Series : Monografias em ciencia da computação.
Nº 14/82. Series Editor: M.A. Casanova. 25p.

* WAGNER, H.J.
1981 The minimum number of mutations in an evolutionary network.
J. theor. Biol. 91: 621-636.

[WAGNER, W.H.
1961 Problems in classification of ferns.
in Recent advances in botany.
Toronto, Univ. Toronto Press : 841-844.]
cité dans Farris, 1970.

WATROUS, L.E. & Q.D. WHEELER
1981 The out-group comparison method of character analysis.
Syst. Zool. : 30(1) : 1-11.

WILEY, E.O.
1981 Phylogenetics.
The theory and practice of phylogenetic systematics.
New York. Chichester. Brisbane. Toronto.
Singapore, A Wiley-interscience publication.
John Wiley & Sons, 439 p.

[WILEY, E.O. & D.R. BROOKS
1982 Victims of history -
A nonequilibrium approach to evolution.
Syst. Zool. 31 : 1-24.]
cité dans Platnick, 1986.

ANNEXE I : Vocabulaire emprunté à la théorie des graphes.
(adapté de Barthélemy et Guénoche 1988).

I.1.La notion de graphe.

"Formellement, un graphe est un couple G = (S,A) formé d'un ensemble fini S = {s,t,u,v,...} et d'un ensemble A de parties à deux éléments de S.

Les éléments de S sont appelés les sommets de G et les éléments de A ses arêtes

Par convention, l'arête {u,v} sera simplement notée uv et on dira que u et v sont adjacents.(...)

L'ensemble des sommets adjacents à un sommet u est appelé le voisinage de u ...

Le nombre d(u) des éléments 'du voisinage de u' est appelé le degré du sommet u ..." (Barthélemy et Guénoche 1988, p.1-2.) Si d(u) = 0, u sera conventionnellement qualifié de sommet isolé.

Un chemin de G entre les sommets u et u' est une suite c: $u=u_0u_1u_2...u_{p-1}u_p=u'$ telle que, pour $0 \leq i \leq p-1$, $u_iu_{i+1} \in A$. De plus, on suppose que chaque arête u_iu_{i+1} n'intervient qu'une fois dans c ...

Un cycle est un chemin dont les extrémités coïncident ...

Le graphe G est dit connexe lorsqu'il existe au moins un chemin entre chaque paire de sommets de G ...

Si A' est un sous ensemble des arêtes du graphe G = (S,A), le graphe G' = (S,A') est appelé le graphe partiel de G induit par A' ...

Un graphe valué est un couple (G,L) formé d'un graphe G = (S,A) et d'une fonction L, à valeurs réelles strictement positives, définie sur A. L(uv) est appelée la longueur... de l'arête uv ..." (Barthélemy et Guénoche 1988, p. 2 et 3.)

I.2. Les arbres.

"DEFINITION 1 : Un arbre est un graphe connexe et sans cycle. Dans un arbre tout sommet de degré 1 est appelé une feuille. Les autres sommets sont appelés ...noeuds. Nous appellerons arbre ternaire un arbre dont tous les noeuds sont de degré 3 ... Un arbre ternaire avec r feuilles possède 2r-2 sommets ..." (Barthélemy et Guénoche, 1988 p. 4 à 5.)

"DEFINITION 2 : Un arbre planté (ou enraciné) "est un couple (H,r) formé d'un arbre H et d'un noeud r de H appelé racine.

Soit (H,r) un arbre planté et soient s,s' deux sommets de H; on pose : s ≤ s' si et seulement si s' est sur le chemin entre s et r.
Il est clair que la relation ≤ est :
- réflexive : pour tout s, s ≤ s ;
- antisymétrique : pour tout s et s',s ≤ s' et s'≤ s impliquent s=s';
- transitive : pour tout s,u,v , s ≤ u et u ≤ v impliquent s ≤ v.

La relation ≤ est donc une relation d'ordre sur l'ensemble des sommets de l'arbre H. Cet ordre admet r pour plus grand élément : pour tout sommet s, s ≤ r ...

Dans un arbre planté (H,r), on dit que l'arête uv est issue du sommet u lorsque v ≤ u, dans ce cas, par analogie avec les arbres généalogiques, on dit que u est le père(...) de v et que v est fils de u. Deux fils d'un même père sont appelés frères..." (Barthélemy et Guénoche, 1988 p. 6.)
Par convention, on dira (lors de l'exposé de l'algorithme de MIX) que la racine d'un arbre est de rang 0, les fils de r de rang 1 et ainsi de suite ...

"DEFINITION 3 : soit X un ensemble fini. Un X-arbre est un couple (H,f), formé d'un arbre H = (S,A) et d'une fonction f de X dans S telle que pour tout v ∈ S-f(X), d(v) ≥ 3.
La fonction f est l'étiquetage du X-arbre. Les sommets dans f(X) sont appelés sommets réels, les sommets dans S-f(X) sont appelés sommets latents" (Barthélemy et Guénoche 1988, p.16.)
"... L'idée est que, compte tenu de données portant sur les sommets réels, les sommets latents jouent le rôle d' 'intermédiaires' nécessaires pour obtenir une structure d'arbre." (Barthélemy et Guénoche 1988, p.13.)

I.3. Arbre minimum dans un graphe valué.

"Soit G un graphe, les deux assertions (i) et (ii) ci-dessous sont équivalentes :
(i) G est connexe
(ii) G admet un graphe partiel qui est un arbre ...

Lorsque le graphe partiel H de G est un arbre, on dit que H est un arbre de G ...

Considérons maintenant un graphe valué connexe (G,L). On définit la longueur L(H) d'un arbre H de G comme la somme des longueurs des arêtes qui le composent. On appelle arbre minimum de (G,L), un arbre H de G tel que L(H) soit minimum" (Barthélemy et Guénoche, 1988 p.10)

Remarques :

1. Au sens de la théorie des graphes (selon les définitions données par Barthélemy et Guénoche 1988), un arbre enraciné dont tous les noeuds sont de degré =3 sauf la racine qui est de degré =2, correspond à l'arbre binaire AB au sens algorithmique tel que défini dans l'annexe II. (voir aussi remarque 3 ci-dessous).
   Dans la description de MIX, seront utilisés les termes arbre binaire pour désigner ce type d'arbre (par opposition aux arbres ternaires tels que définis par Barthélemy et Guénoche 1988) et de racine pour qualifier un noeud de degré 2.
   Comme définit ci-dessus, un arbre binaire AB avec spp feuilles possède donc 2spp-1 sommets.
   Cette définition est implicitement acceptée par Felsenstein lui-même dans les commentaires de certaines procédures de MIX (:"constructs a binary tree ...").
   Dans la description de l'algorithme de MIX, un arbre binaire ayant, à la suite d'un processus de construction pas à pas, incorporé ces 2spp-1 sommets sera qualifié de "complété". D'autre part un sommet incorporé à l'arbre binaire AB sera dit "lié" à AB.

2. Il existe de nombreuses divergences au niveau du vocabulaire utilisé par les biologistes systématiciens et les mathématiciens. C'est ainsi que Swofford et Berlocher (1987 p. 294) nomment arbre binaire non-enraciné ("undirected or unrooted binary tree") ce qui est ici appelé "arbre ternaire non-enraciné." Ce dernier est appelé "réseau" (network) par Farris (1970) et beaucoup d'autres (Wagner 1981) qui réservent le terme d'arbre ("tree") à l'arbre enraciné. Un arbre dont tous les noeuds sont de degré trois est parfois qualifié dans la littérature anglosaxone de "bifurcating tree" (Graham & Foulds 1982, p.134) ou encore "fully bifurcating tree" (Berlocher and Swofford, 1987 p.298). Il est souvent nommé binaire (Day, 1983, p.432).

3. Il est éclairant de noter que le vocabulaire emprunté à la théorie des graphes s'inscrit dans une sémantique déclarative.

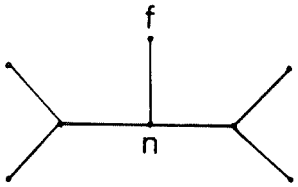


Figure I.1 : Un arbre ternaire.
n = noeud
f=feuille.

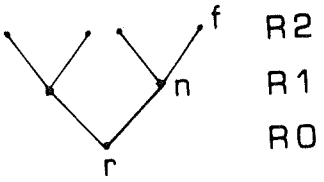


Figure I.2 : Un arbre binaire.
n = noeud
f = feuille
r = racine
R = rang.

ANNEXE II : Vocabulaire emprunté à l'algorithmique
(cf. cours intitulé "Méthodologie de la programmation du Professeur H.Leroy).

Arbre Binaire AB de type T : ensemble de variables de type T structuré selon la définition suivante :

- vide
- formé de 3 parties : .racine (variable de type T)
  .sous-arbre de gauche (Arbre Binaire de type T)
  .sous-arbre de droite (Arbre Binaire de type T)

Parcours de AB en ordre préfixé(préordre) :

- si AB est vide : ne rien faire
- sinon 1.visiter la racine
  2.parcourir le sous-arbre de gauche en ordre préfixé
  3.parcourir le sous-arbre de droite en ordre préfixé.

Parcours de AB en ordre postfixé(postordre) :

- si AB est vide : ne rien faire
- sinon 1.parcourir le sous-arbre de gauche en ordre postfixé
  2.parcourir le sous-arbre de droite en ordre postfixé
  3.visiter la racine.

Remarque : Ce vocabulaire, contrairement à celui emprunté à la théorie des graphes, relève d'une sémantique procédurale.

ANNEXE - III
PROGRAMME MIX.PAS
(Extrait de Phylip 3.1 de Felsenstein)

```
PROGRAM Mix (infile, treefile, OUTPUT);
 (* version 3.1. (c) Copyright 1986, 1987, 1988 by the University of
    Washington and Joseph Felsenstein.  Written by Joseph Felsenstein.
    Permission is granted to copy and use this program provided no
    fee is charged for it and provided that this copyright
    notice is not removed. *)

CONST maxsp = 50;    (* maximum number of species              *)
  maxsz = 100;      (* size of pointer array.  >= 2*maxsp - 1  *)
  maxchr = 200;     (* max. number characterstates per species *)
  bits = 30;        (* max. number of elements in a Pascal set *)
  wrds = 7;         (* wrds * bits >= maxchr                   *)
  nmlngth = 10;     (* number of characters in species name    *)
  maxtrees = 100;   (* maximum number of tied trees stored     *)
  maxuser = 10;     (* maximum number of user-defined trees    *)

TYPE steparray = ARRAY[1..maxchr] OF INTEGER;
  bitrange = 1..bits;
  bitset = SET OF bitrange;
  bitarray = ARRAY[1..wrds] OF bitset;
  nodeptr = ^node;  (* nodes will form a binary tree *)
  node = RECORD     (* describes a tip species or an ancestor *)
    ancestor, leftdesc, rtdesc : nodeptr;  (* pointers to nodes *)
    index : INTEGER;      (* number of the node *)
    tip : BOOLEAN;        (* present species are tips of tree *)
    stateone, statezero : bitarray; (* see in PROCEDURE fillin *)
    xcoord, ycoord : INTEGER    (* used by printree *)
    END;
  pointarray = ARRAY[1..maxsz] OF nodeptr;
  longer = ARRAY[0..5] OF INTEGER;

VAR root : nodeptr;
  infile, treefile : TEXT;
  spp, nonodes, chars, words, inseed, outgrno : INTEGER;
   (* spp = number of species
      nonodes = number of nodes in tree
      chars = number of binary characters
      words = number of words needed
            to represent characters of one organism
      outgrno indicates outgroup *)
  jumble, usertree, weights, thresh, ancvar, questions, allsokal,
    allwagner, mixture, trout, noroot, outgropt, didreroot : BOOLEAN;
  extras, weight : steparray;
  ancone, anczero : ARRAY[1..maxchr] OF BOOLEAN;
  treenode : pointarray;      (* pointers to all nodes in tree *)
  name : ARRAY[1..maxsp, 1..nmlngth] OF CHAR; (* names of species *)
  threshold : REAL;
  threshwt : ARRAY[1..maxchr] OF REAL;
  wagner : bitarray;
  seed : longer;
  enterorder : ARRAY[1..maxsp] OF INTEGER;

  FUNCTION randum (VAR seed : longer) : REAL;
    (* random number generator -- slow but machine independent *)
    VAR i, j, k, sum : INTEGER;
      mult, newseed : longer;
```

```
    x : REAL;
BEGIN (* randum *)
  mult[0] := 13;
  mult[1] := 24;
  mult[2] := 22;
  mult[3] :=  6;
  FOR i := 0 TO 5 DO
    newseed[i] := 0;
  FOR i := 0 TO 5 DO BEGIN
    sum := newseed[i];
    k := i;
    IF i > 3 THEN k := 3;
    FOR j := 0 TO k DO
      sum := sum + mult[j] * seed[i-j];
    newseed[i] := sum;
    FOR j := i TO 4 DO BEGIN
     newseed[j+1] := newseed[j+1] + newseed[j] DIV 64;
     newseed[j] := newseed[j] MOD 64;
      END;
    END;
  seed := newseed;
  seed[5] := seed[5] MOD 4;
  x := 0.0;
  FOR i := 0 TO 5 DO
    x := x / 64.0 + seed[i];
  x := x / 4.0;
  randum := x;
END; (* randum *)

FUNCTION digit (ch : CHAR) : BOOLEAN;
BEGIN (* digit *)
  digit := (ch >= '0') AND (ch <= '9');
END; (* digit *)

FUNCTION letter (ch : CHAR) : BOOLEAN;
  (* tests whether ch is a letter
    -- works for both ASCII and EBCDIC codes *)
BEGIN (* letter *)
  letter := ((ch >= 'A') AND (ch <= 'I'))
          OR ((ch >= 'J') AND (ch <= 'R'))
          OR ((ch >= 'S') AND (ch <= 'Z'))
          OR ((ch >= 'a') AND (ch <= 'i'))
          OR ((ch >= 'j') AND (ch <= 'r'))
          OR ((ch >= 's') AND (ch <= 'z'));
END; (* letter *)

PROCEDURE uppercase (VAR ch : CHAR);
  (* convert a character to upper case -- either ASCII or EBCDIC *)
BEGIN (* uppercase *)
  IF letter(ch) AND (('a' > 'A') AND (ch >= 'a')
              OR  ('a' < 'A') AND (ch < 'A'))
  THEN ch := CHR(ORD(ch) + ORD('A') - ORD('a'));
END; (* uppercase *)

PROCEDURE newline (i, j, k : INTEGER);
  (* go to new line if i is a multiple of j, indent k spaces *)
  VAR m : INTEGER;
BEGIN (* newline *)
  IF (((i-1) MOD j) = 0) AND (i>1)
  THEN BEGIN
    WRITELN(OUTPUT);
    FOR m := 1 TO k DO WRITE(OUTPUT, ' ');
    END;
END; (* newline *)
```

```
PROCEDURE doinput;
  (* reads the input data *)

  PROCEDURE inputnumbers;
    (* input the numbers of species and of characters *)
  BEGIN (* inputnumbers *)
    WRITELN(OUTPUT);
    WRITELN(OUTPUT, ' Mixed parsimony algorithm, version 3.1');
    WRITELN(OUTPUT);
    READ(infile, spp, chars);
    WRITELN(OUTPUT, ' ',spp:2,' species, ',chars:3,'  characters');
    IF spp > maxsp
    THEN WRITELN(OUTPUT, ' TOO MANY SPECIES: adjust CONSTants');
    IF chars > maxchr
    THEN WRITELN(OUTPUT, ' TOO MANY CHARACTERS: adjust CONSTants');
    WRITELN(OUTPUT);
    words := (chars DIV bits) + 1;
    nonodes := 2 * spp - 1;
  END; (* inputnumbers *)

  PROCEDURE inputoptions;
    (* input the information on the options *)
    VAR ch : CHAR;
      extranum, i : INTEGER;

    PROCEDURE inputmixture;
      (* input mixture of methods *)
      VAR i, j, k : INTEGER;
        ch : CHAR;
        wag : BOOLEAN;
    BEGIN (* inputmixture *)
      FOR i := 1 TO nmlngth-1 DO READ(infile, ch);
      FOR i := 1 TO words DO wagner[i] := [];
      j := 0;
      k := 1;
      FOR i := 1 TO chars DO BEGIN
        REPEAT
          IF EOLN(infile) THEN READLN(infile);
          READ(infile, ch);
        UNTIL ch <> ' ';
        uppercase (ch);
        wag := FALSE;
        IF (ch='W') OR (ch = '?')
        THEN wag := TRUE
        ELSE IF (ch='S') OR (ch='C')
          THEN wag := FALSE
          ELSE WRITELN(OUTPUT, ' error: ', ch);
        j := j + 1;
        IF j > bits
        THEN BEGIN
          j := 1;
          k := k + 1;
          END;
        IF wag
        THEN wagner[k] := wagner[k] + [j];
        IF NOT ancvar
        THEN ancone[i] := wag;
        END;
      READLN(infile);
      allwagner := FALSE;
      allsokal := FALSE;
      mixture := TRUE;
    END; (* inputmixture *)

    PROCEDURE printmixture;
```

```
  (* print out list of parsimony methods *)
   VAR i, k, l : INTEGER;
BEGIN (* printmixture *)
  allwagner := TRUE;
  WRITELN(OUTPUT, ' Parsimony methods:');
  l := 0;
  k := 1;
  FOR i := 1 TO nmlngth+4 DO WRITE(OUTPUT, ' ');
  FOR i := 1 TO chars DO BEGIN
     newline (i, 50, nmlngth+4);
     l := l + 1;
     IF l > bits
    THEN BEGIN
       l := 1;
       k := k + 1;
       END;
    IF l IN wagner[k]
    THEN WRITE(OUTPUT, 'W')
    ELSE BEGIN
       WRITE(OUTPUT, 'S');
      allwagner := allwagner AND (weight[i] = 0);
       END;
    IF ((i MOD 5) = 0) THEN WRITE(OUTPUT, ' ');
     END;
  WRITELN(OUTPUT);
  WRITELN(OUTPUT);
END; (* printmixture *)

PROCEDURE inputthresh;
  (* input the threshold *)
BEGIN (* inputthresh *)
  READLN(infile, threshold);
  thresh := TRUE;
END; (* inputthresh *)

PROCEDURE printthresh;
  (* print out the threshold value *)
BEGIN (* printthresh *)
  WRITELN(OUTPUT, '    Threshold value = ', threshold:10:3);
  WRITELN(OUTPUT);
END; (* printthresh *)

PROCEDURE inputweights;
  (* input the character weights, 0-9 and A-Z for weights
     0 - 35 *)
   VAR ch : CHAR;
     i : INTEGER;
BEGIN (* inputweights *)
  FOR i := 1 TO nmlngth-1 DO READ(infile, ch);
  FOR i := 1 TO chars DO BEGIN
     REPEAT
      IF EOLN(infile) THEN READLN(infile);
      READ(infile, ch);
     UNTIL ch <> ' ';
     weight[i] := 1;
     IF digit(ch)
    THEN weight[i] := ORD(ch) - ORD('0')
    ELSE IF letter(ch)
      THEN BEGIN
        uppercase (ch);
        IF (ch >= 'A') AND (ch <= 'I')
        THEN weight[i] := ORD(ch) - ORD('A') + 10
        ELSE IF (ch >= 'J') AND (ch <= 'R')
          THEN weight[i] := ORD(ch) - ORD('J') + 19
          ELSE weight[i] := ORD(ch) - ORD('S') + 28;
```

```
          END
        ELSE WRITELN(OUTPUT, ' Bad weight character: ',ch);
      END;
   READLN(infile);
   weights := TRUE;
END; (* inputweights *)

PROCEDURE printweights;
  (* print out the weights of characters *)
   VAR i, j, k : INTEGER;
BEGIN (* printweights *)
  WRITELN(OUTPUT,'   Characters are weighted as follows:');
   WRITE(OUTPUT, '       ');
   FOR i := 0 TO 9 DO WRITE(OUTPUT, i:3);
   WRITELN(OUTPUT);
 WRITELN(OUTPUT,'    *----------------------------------');
  FOR j := 0 TO (chars DIV 10) DO BEGIN
     WRITE(OUTPUT, 10*j:5, '!  ');
     FOR i := 0 TO 9 DO BEGIN
        k := 10*j + i;
       IF (k>0) AND (k<=chars)
       THEN WRITE(OUTPUT, weight[k]:3)
       ELSE WRITE(OUTPUT, '   ');
       END;
     WRITELN(OUTPUT);
     END;
   WRITELN(OUTPUT);
END; (* printweights *)

PROCEDURE inputancestors;
 (* reads the ancestral states for each character *)
   VAR i : INTEGER;
      ch : CHAR;
BEGIN (* inputancestors *)
   FOR i := 1 TO nmlngth-1 DO READ(infile, ch);
   ancvar := TRUE;
   FOR i := 1 TO chars DO BEGIN
      REPEAT
        IF EOLN(infile) THEN READLN(infile);
        READ(infile, ch);
      UNTIL ch <> ' ';
      IF ch = 'p' THEN ch := 'P';
      IF ch = 'b' THEN ch := 'B';
      CASE ch OF
        '1' : anczero[i] := FALSE;
        '0' : ancone[i] := FALSE;
        'P' : BEGIN END;
        'B' : BEGIN END;
        '?' : BEGIN END
        END;
      END;
   READLN(infile);
END; (* inputancestors *)

PROCEDURE printancestors;
   (* print out list of ancestral states *)
   VAR i : INTEGER;
BEGIN (* printancestors *)
   WRITELN(OUTPUT, ' Ancestral states:');
   FOR i := 1 TO nmlngth+4 DO WRITE(OUTPUT, ' ');
   FOR i := 1 TO chars DO BEGIN
      newline (i, 50, nmlngth+4);
     IF ancone[i] AND anczero[i]
     THEN WRITE(OUTPUT, '?')
     ELSE IF ancone[i]
```

```
          THEN WRITE(OUTPUT, '1')
          ELSE WRITE(OUTPUT, '0');
        IF ((i MOD 5) = 0) THEN WRITE(OUTPUT, ' ');
        END;
      WRITELN(OUTPUT);
      WRITELN(OUTPUT);
    END; (* printancestor *)

BEGIN (* inputoptions *)
  ancvar := FALSE;
  FOR i := 1 TO chars DO BEGIN
    anczero[i] := TRUE;
    ancone[i] := TRUE;
    END;
  allwagner := TRUE;
  allsokal := FALSE;
  jumble := FALSE;
  mixture := FALSE;
  outgrno := 1;
  outgropt := FALSE;
  thresh := FALSE;
  trout := FALSE;
  usertree := FALSE;
  weights := FALSE;
  extranum := 0;
  WHILE NOT EOLN(infile) DO BEGIN
    READ(infile, ch);
    uppercase (ch);
    IF (ch = 'A') OR (ch = 'C') OR (ch = 'J')
      OR (ch = 'M') OR (ch = 'O') OR (ch = 'S')
      OR (ch = 'T') OR (ch = 'U') OR (ch = 'W') OR (ch = 'Y')
    THEN CASE ch OF
        'A' : extranum := extranum + 1;
        'C' : allsokal := TRUE;
        'J' : BEGIN
              jumble := TRUE;
              extranum := extranum + 1;
              END;
        'M' : extranum := extranum + 1;
        'O' : BEGIN
              outgropt := TRUE;
              extranum := extranum + 1;
              END;
        'S' : allsokal := TRUE;
        'T' : extranum := extranum + 1;
        'U' : usertree := TRUE;
        'W' : extranum := extranum + 1;
        'Y' : trout := TRUE
        END
    ELSE IF ch <> ' '
      THEN WRITELN(OUTPUT, ' Bad Option Character: ', ch);
    END;
  READLN(infile);
  IF allsokal THEN allwagner := FALSE;
  FOR i := 1 TO words DO
    IF allsokal
    THEN wagner[i] := []
    ELSE wagner[i] := [1..bits];
  threshold := spp;
  FOR i := 1 TO chars DO weight[i] := 1;
  FOR i := 1 TO extranum DO BEGIN
    READ(infile, ch);
    uppercase (ch);
    IF ch = 'A'
    THEN inputancestors;
```

```
      IF ch = 'J'
      THEN READLN(infile, inseed);
      IF ch = 'M'
      THEN inputmixture;
      IF ch = 'O'
      THEN READLN(infile, outgrno);
      IF ch = 'T'
      THEN inputthresh;
      IF ch = 'W'
      THEN inputweights;
      END;
    IF allsokal
    THEN BEGIN
      IF NOT ancvar
      THEN FOR i := 1 TO chars DO ancone[i] := FALSE;
      WRITELN(OUTPUT, ' Camin-Sokal parsimony method');
      END;
    IF allwagner
    THEN WRITELN(OUTPUT, ' Wagner parsimony method');
    WRITELN(OUTPUT);
    IF mixture
    THEN printmixture;
    IF jumble
    THEN BEGIN
      WRITELN(OUTPUT, ' Random number seed = ', inseed:8);
      WRITELN(OUTPUT);
      FOR i := 0 TO 5 DO seed[i] := 0;
      i := 0;
      REPEAT
        seed[i] := inseed MOD 64;
        inseed := inseed DIV 64;
        i := i + 1;
      UNTIL inseed = 0;
      END;
    IF thresh
    THEN printthresh;
    IF weights
    THEN printweights;
    IF ancvar
    THEN printancestors;
    noroot := allwagner OR (threshold <= 2.0);
    questions := FALSE;
    FOR i := 1 TO chars DO BEGIN
      noroot := noroot AND ancone[i] AND anczero[i];
      questions := questions OR (ancone[i] AND anczero[i]);
      threshwt[i] := threshold * weight[i];
      END;
  END; (* inputoptions *)

  PROCEDURE inputdata;
    (* input the names and character state data for species *)
    VAR i, j, l : INTEGER;
      k : bitrange;
      charstate : CHAR;  (* possible states are
                          '0', '1', 'P', 'B', and '?' *)
  BEGIN (* inputdata *)
    WRITELN(OUTPUT);
    j := nmlngth - 5 + chars DIV 2;
    IF j < nmlngth - 1 THEN j := nmlngth - 1;
    IF j > 39 THEN j := 39;
    WRITE(OUTPUT, ' Name');
    FOR i := 1 TO j DO WRITE(OUTPUT, ' ');
    WRITELN(OUTPUT, 'Characters');
    WRITE(OUTPUT, ' ----');
    FOR i := 1 TO j DO WRITE(OUTPUT, ' ');
```

```
    WRITELN(OUTPUT, '----------');
    WRITELN(OUTPUT);
    FOR i := 1 TO chars DO extras[i] := 0;
    FOR i := 1 TO nonodes DO BEGIN
      NEW (treenode[i]);
      treenode[i]^.leftdesc := NIL;
      treenode[i]^.rtdesc := NIL;
      treenode[i]^.ancestor := NIL;
      treenode[i]^.index := i;
       IF i <= spp
       THEN BEGIN
         treenode[i]^.tip := TRUE;
         FOR j := 1 TO nmlngth DO READ(infile, name[i,j]);
         WRITE(OUTPUT, ' ');
         FOR j := 1 TO nmlngth DO WRITE(OUTPUT, name[i,j]);
         WRITE(OUTPUT, '   ');
         FOR j := 1 TO words DO BEGIN
          treenode[i]^.stateone[j] := [];
          treenode[i]^.statezero[j] := [];
           END;
         FOR j := 1 TO chars DO BEGIN
           k := ((j-1) MOD bits) + 1;
           l := ((j-1) DIV bits) + 1;
           REPEAT
             IF EOLN(infile) THEN READLN(infile);
             READ(infile, charstate);
           UNTIL charstate <> ' ';
          IF charstate = 'b' THEN charstate := 'B';
          IF charstate = 'p' THEN charstate := 'P';
          IF NOT ((charstate = '0')
                OR (charstate = '1')
                OR (charstate = '?')
                OR (charstate = 'P')
                OR (charstate = 'B'))
          THEN WRITELN(OUTPUT, ' Warning -- ',
             'bad character state: ', charstate,
              ' at character ', j:5, ' of species ', i:3);
           newline (j, 60, nmlngth+4);
          WRITE(OUTPUT, charstate);
          IF (j MOD 5) = 0 THEN WRITE(OUTPUT, ' ');
          IF (charstate = '1')
         THEN treenode[i]^.stateone[l]
                  := treenode[i]^.stateone[l] + [k];
          IF (charstate = '0')
         THEN treenode[i]^.statezero[l]
                  := treenode[i]^.statezero[l] + [k];
          IF (charstate = 'P') OR (charstate = 'B')
          THEN extras[j] := extras[j] + weight[j];
           END;
         READLN(infile);
         WRITELN(OUTPUT);
         END
       ELSE treenode[i]^.tip := FALSE;
        END;
     WRITELN(OUTPUT);
     WRITELN(OUTPUT);
  END; (* inputdata *)

BEGIN (* doinput *)
   inputnumbers;
   inputoptions;
   inputdata;
END; (* doinput *)

PROCEDURE maketree;
```

```
(* constructs a binary tree from the pointers in treenode.
  adds each node at location which yields highest "likelihood"
  then rearranges the tree for greatest "likelihood" *)
TYPE placearray = ARRAY[1..maxsz] OF INTEGER;
 VAR i, j, k, nextree, numtrees, which, minwhich : INTEGER;
    like, bestyet, bestlike, gotlike,
    minsteps, sum, sum2, sd : REAL;
   lastrearr : BOOLEAN;
    nsteps : ARRAY [1..maxuser] OF REAL;
    fsteps : ARRAY [1..maxuser] OF ARRAY [1..maxchr] OF REAL;
    guess : ARRAY[1..maxchr] OF CHAR;
    there, item, nufork, dummy : nodeptr;
    numsteps : steparray;
    fullset : bitset;
   steps, zeroanc, oneanc : bitarray;
   place : placearray;
   bestrees : ARRAY[1..maxtrees] OF placearray;

PROCEDURE add (below, newtip, newfork : nodeptr);
  (* inserts the nodes newfork and its left descendant, newtip,
     to the tree.  below becomes newfork's right descendant *)
BEGIN (* add *)
  newfork^.ancestor := below^.ancestor;
  newfork^.rtdesc := below;
  newfork^.leftdesc := newtip;
  below^.ancestor := newfork;
  newtip^.ancestor := newfork;
  IF newfork^.ancestor <> NIL
  THEN IF newfork^.ancestor^.leftdesc = below
    THEN newfork^.ancestor^.leftdesc := newfork
    ELSE newfork^.ancestor^.rtdesc := newfork;
END; (* add *)

PROCEDURE remove (VAR item, fork : nodeptr);
  (* removes nodes item and its ancestor, fork, from the tree.
     the new descendant of fork's ancestor is made to be
     fork's second descendant (other than item).  Also
     returns pointers to the deleted nodes, item and fork *)
   VAR other, below : nodeptr;
BEGIN (* remove *)
  IF item^.ancestor = NIL
  THEN fork := NIL
  ELSE BEGIN
     fork := item^.ancestor;
     IF item = fork^.leftdesc
     THEN other := fork^.rtdesc
     ELSE other := fork^.leftdesc;
     below := fork^.ancestor;
     other^.ancestor := below;
      IF below <> NIL
     THEN BEGIN
       IF below^.leftdesc = fork
       THEN below^.leftdesc := other
       ELSE below^.rtdesc := other;
       fork^.ancestor := NIL;
        END;
    fork^.leftdesc := NIL;
    fork^.rtdesc := NIL;
    item^.ancestor := NIL;
     END;
END; (* remove *)

PROCEDURE fillin (p: nodeptr);
  (* Sets up for each node in the tree two statesets.
    stateone and statezero are the sets of character
```

```
      states that must be 1 or must be 0, respectively,
      in a most parsimonious reconstruction, based on the
      information at or above this node.  Note that this
      state assignment may change based on information further
      down the tree.  If a character is in both sets it is in
      state "P".  If in neither, it is "?". *)
    VAR i : INTEGER;
        l0, l1, r0, r1, st, wa, za : bitset;
BEGIN (* fillin *)
  FOR i := 1 TO words DO BEGIN
    l0 := p^.leftdesc^.statezero[i];
    l1 := p^.leftdesc^.stateone[i];
    r0 := p^.rtdesc^.statezero[i];
    r1 := p^.rtdesc^.stateone[i];
     wa := wagner[i];
     za := zeroanc[i];
      st := l1 * r0 + l0 * r1;
      steps[i] := st;
     p^.stateone[i] := l1 + r1 - (st * (wa + za));
     p^.statezero[i] := l0 + r0 - (st * (wa + (fullset - za)));
      END;
END; (* fillin *)

PROCEDURE evaluate (r: nodeptr);
  (* Determines the number of steps needed for a tree.
     This is the minimum number needed to evolve chars on
     this tree *)
   VAR i, stepnum, smaller : INTEGER;
      sum, term : REAL;
      numsone, numszero : steparray;

   PROCEDURE count (VAR stps : bitarray);
     (* counts the number of steps in a fork of the tree.
        The program spends much of its time in this PROCEDURE *)
      VAR i, j, l : INTEGER;
   BEGIN (* count *)
     j := 1;
     l := 0;
     FOR i := 1 TO chars DO BEGIN
        l := l + 1;
        IF l > bits
        THEN BEGIN
           l := 1;
           j := j + 1;
           END;
        IF l IN stps[j]
        THEN BEGIN
          IF l IN zeroanc[j]
          THEN numszero[i] := numszero[i] + weight[i]
          ELSE numsone[i] := numsone[i] + weight[i];
          END;
        END;
   END; (* count *)

   PROCEDURE postorder (p : nodeptr);
     (* traverses a binary tree, calling PROCEDURE fillin at a
        node's descendants before calling fillin at the node *)
   BEGIN (* postorder *)
     IF NOT p^.tip
     THEN BEGIN
       postorder (p^.leftdesc);
       postorder (p^.rtdesc);
        fillin (p);
        count (steps);
        END;
```

```
  END; (* postorder *)

BEGIN (* evaluate *)
   sum := 0;
  FOR i := 1 TO chars DO BEGIN
     numszero[i] := 0;
     numsone[i] := 0;
     END;
  FOR i := 1 TO words DO
    zeroanc[i] := fullset;
  postorder (r);
  count (r^.stateone);
  FOR i := 1 TO words DO
    zeroanc[i] := [];
  postorder (r);
  count (r^.statezero);
  FOR i := 1 TO chars DO BEGIN
     smaller := spp * weight[i];
     IF anczero[i]
     THEN BEGIN
       numsteps[i] := numszero[i];
       smaller := numszero[i];
       END;
     IF ancone[i] AND (numsone[i] < smaller)
     THEN numsteps[i] := numsone[i];
     stepnum := numsteps[i] + extras[i];
     IF stepnum <= threshwt[i]
     THEN term := stepnum
     ELSE term := threshwt[i];
     sum := sum + term;
     IF usertree
     THEN fsteps[which][i] := term;
     guess[i] := '?';
     IF (NOT ancone[i]) OR
       (anczero[i] AND (numszero[i] < numsone[i]))
     THEN guess[i] := '0'
     ELSE IF (NOT anczero[i]) OR
          (ancone[i] AND (numsone[i] < numszero[i]))
       THEN guess[i] := '1';
     END;
  IF usertree
  THEN BEGIN
     nsteps[which] := sum;
     IF which = 1
     THEN BEGIN
       minwhich := 1;
       minsteps := sum;
       END
     ELSE IF sum < minsteps
       THEN BEGIN
         minwhich := which;
         minsteps := sum;
         END;
     END;
   like := -sum;
END; (* evaluate *)

PROCEDURE reroot (outgroup : nodeptr);
  (* reorients tree, putting outgroup in desired position.
     Written by Kent Fiala *)
  VAR vertex, oldvertex, ante, otherbranch : nodeptr;
     left : BOOLEAN;
BEGIN (* reroot *)
  IF NOT (outgroup^.ancestor = root)
  THEN BEGIN
```

```
      oldvertex := outgroup;
      vertex := outgroup^.ancestor;
      ante := vertex^.ancestor;
      IF vertex^.rtdesc = outgroup
       THEN BEGIN
        vertex^.rtdesc := vertex^.leftdesc;
         vertex^.leftdesc := outgroup;
          END;
       REPEAT
        IF oldvertex = vertex^.leftdesc
         THEN BEGIN
           vertex^.leftdesc := ante;
            left := TRUE;
            END
         ELSE BEGIN
           vertex^.rtdesc := ante;
            left := FALSE;
            END;
        vertex^.ancestor := oldvertex;
        oldvertex := vertex;
        vertex := ante;
       ante := vertex^.ancestor
      UNTIL vertex = root;
     IF root^.leftdesc = oldvertex
     THEN otherbranch := root^.rtdesc
     ELSE otherbranch := root^.leftdesc;
      IF left
     THEN oldvertex^.leftdesc := otherbranch
     ELSE oldvertex^.rtdesc := otherbranch;
     otherbranch^.ancestor := oldvertex;
     root^.rtdesc := outgroup^.ancestor;
     root^.leftdesc := outgroup;
     outgroup^.ancestor^.ancestor := root;
     outgroup^.ancestor := root;
      END;
END; (* reroot *)

PROCEDURE addpreorder (p, item, nufork : nodeptr);
  (* traverses a binary tree, calling PROCEDURE tryadd
    at a node before calling tryadd at its descendants *)
  TYPE knowed = RECORD    (* temporary storage for tree shape *)
    ancestor, leftdesc, rtdesc : nodeptr;
     END;
  VAR treeknowed : ARRAY [1..maxsz] OF knowed;

  PROCEDURE tryadd (p: nodeptr);
    (* temporarily adds one fork and one tip to the tree.
      if the location where they are added yields greater
       "likelihood" than other locations tested up to that
       time, then keeps that location as there *)
     VAR i, pos : INTEGER;
       found : BOOLEAN;
       rute : nodeptr;

     PROCEDURE savetree;
       (* record in place where each species has to be
          added to reconstruct this tree *)
        VAR i, j : INTEGER;
          p : nodeptr;
          done : BOOLEAN;
     BEGIN (* savetree *)
        IF noroot
       THEN reroot (treenode[outgrno]);
       FOR i := 1 TO nonodes DO place[i] := 0;
       place[root^.index] := 1;
```

```
    FOR i := 1 TO spp DO BEGIN
      p := treenode[i];
     WHILE (place[p^.index] = 0) DO BEGIN
        place[p^.index] := i;
        p := p^.ancestor;
       END;
      IF i > 1
      THEN BEGIN
       place[i] := place[p^.index];
       j := place[p^.index];
        done := FALSE;
       WHILE NOT done DO BEGIN
          place[p^.index] := spp + i - 1;
          p := p^.ancestor;
          done := (p = NIL);
          IF NOT done
         THEN done := (place[p^.index] <> j);
          END;
        END;
      END;
  END; (* savetree *)

  PROCEDURE findtree;
    (* finds tree given by ARRAY place in ARRAY
      bestrees by binary search *)
    VAR i, lower, upper : INTEGER;
      below : BOOLEAN;
  BEGIN (* findtree *)
    below := FALSE;
    lower := 1;
    upper := nextree - 1;
    found := FALSE;
   WHILE (NOT found) AND (lower <= upper) DO BEGIN
      pos := (lower + upper) DIV 2;
       i := 3;
     WHILE (place[i] = bestrees[pos][i]) AND (i <= spp) DO
         i := i + 1;
      found := (i > spp);
     below := place[i] < bestrees[pos][i];
      IF NOT found
      THEN BEGIN
        IF below
       THEN upper := pos - 1
        ELSE lower := pos + 1;
        END;
      END;
    IF (NOT found) AND (NOT below)
     THEN pos := pos + 1;
  END; (* findtree *)

  PROCEDURE addtree;
    (* puts tree from ARRAY place in its proper position
      in ARRAY bestrees *)
    VAR i : INTEGER;
  BEGIN (* addtree *)
    FOR i := (nextree - 1) DOWNTO pos DO
      bestrees[i+1] := bestrees[i];
    bestrees[pos] := place;
    nextree := nextree + 1;
  END; (* addtree *)

BEGIN (* tryadd *)
  add (p, item, nufork);
  IF root^.ancestor <> NIL THEN root := root^.ancestor;
  evaluate (root);
```

```
      IF lastrearr
      THEN BEGIN
         IF like >= bestlike
         THEN BEGIN
           FOR i := 1 TO nonodes DO BEGIN
            treeknowed[i].ancestor := treenode[i]^.ancestor;
            treeknowed[i].leftdesc := treenode[i]^.leftdesc;;
            treeknowed[i].rtdesc := treenode[i]^.rtdesc;
              END;
           rute := root;
           savetree;
           FOR i := 1 TO nonodes DO BEGIN
            treenode[i]^.ancestor := treeknowed[i].ancestor;
            treenode[i]^.leftdesc := treeknowed[i].leftdesc;;
            treenode[i]^.rtdesc := treeknowed[i].rtdesc;
              END;
           root := rute;
           IF like > bestlike
           THEN BEGIN
              bestlike := like;
              pos := 1;
              nextree := 1;
             addtree;
              END
           ELSE BEGIN
              pos := 0;
             findtree;
             IF NOT found
             THEN IF nextree <= maxtrees
               THEN addtree;
             END;
          END;
        IF like > bestyet
        THEN BEGIN
          bestyet := like;
          there := p;
          END;
        remove (item, nufork);
        IF nufork = root THEN root := p;
    END; (* tryadd *)

BEGIN (* addpreorder *)
     IF p <> NIL
     THEN BEGIN
       tryadd (p);
      addpreorder (p^.leftdesc, item, nufork);
      addpreorder (p^.rtdesc, item, nufork);
       END;
END; (* addpreorder *)

PROCEDURE rearrange (VAR r: nodeptr);
   (* traverses the tree (preorder), finding any local
      rearrangement which decreases the number of steps.
      if traversal succeeds in increasing the tree's
      "likelihood", PROCEDURE rearrange runs traversal again *)
    VAR success : BOOLEAN;

    PROCEDURE tryrearr (p : nodeptr);
      (* evaluates one rearrangement of the tree.
         if the new tree has greater "likelihood" than the old
         one sets success := TRUE and keeps the new tree.
         otherwise, restores the old tree *)
      VAR frombelow, whereto, forknode : nodeptr;
          oldlike : REAL;
```

```
  BEGIN (* tryrearr *)
    IF p^.ancestor <> NIL
    THEN BEGIN
      IF p^.ancestor^.ancestor <> NIL
      THEN BEGIN
        oldlike := bestyet;
       IF p^.ancestor^.leftdesc = p
       THEN frombelow := p^.ancestor^.rtdesc
       ELSE frombelow := p^.ancestor^.leftdesc;
       whereto := p^.ancestor^.ancestor;
       forknode := p^.ancestor;
       remove (p, forknode);
       add (whereto, p, forknode);
       IF r^.ancestor <> NIL THEN r := r^.ancestor;
       evaluate (r);
       IF like <= oldlike
       THEN BEGIN
         remove (p, forknode);
         add (frombelow, p, forknode);
         IF r^.ancestor <> NIL THEN r := r^.ancestor;
         END
       ELSE BEGIN
         success := TRUE;
         bestyet := like;
         END;
       END;
     END;
  END; (* tryrearr *)

  PROCEDURE repreorder (p : nodeptr);
    (* traverses a binary tree, calling PROCEDURE tryrearr
      at a node before calling tryrearr at its descendants *)
  BEGIN (* repreorder *)
    IF p <> NIL
    THEN BEGIN
      tryrearr (p);
     repreorder (p^.leftdesc);
     repreorder (p^.rtdesc);
      END;
  END; (* repreorder *)

BEGIN (* rearrange *)
  success := TRUE;
  WHILE success DO BEGIN
    success := FALSE;
   repreorder (r);
    END;
END; (* rearrange *)

PROCEDURE treeread;
  (* read in user-defined tree and set it up *)
   VAR ch : CHAR;
     nextnode : INTEGER;

   PROCEDURE findch (c : CHAR);
    (* scan forward until find character c *)
  BEGIN (* findch *)
    IF ch <> c
    THEN REPEAT
        IF EOLN(infile) THEN READLN(infile);
        READ(infile, ch);
      UNTIL ch = c;
  END; (* findch *)

  PROCEDURE addelement (VAR p : nodeptr);
```

```
    (* recursive procedure adds nodes to user-defined tree *)
     VAR q : nodeptr;
        i, n : INTEGER;
       found : BOOLEAN;
        str : ARRAY[1..nmlngth] OF CHAR;
  BEGIN (* addelement *)
     REPEAT
      IF EOLN(infile) THEN READLN(infile);
      READ(infile, ch);
     UNTIL ch <> ' ';
     IF ch = '('
     THEN BEGIN
       nextnode := nextnode + 1;
      q := treenode[nextnode];
      addelement (q^.leftdesc);
      q^.leftdesc^.ancestor := q;
       findch (',');
      addelement (q^.rtdesc);
      q^.rtdesc^.ancestor := q;
       findch (')');
       p := q;
       END
     ELSE BEGIN
       FOR i := 1 TO nmlngth DO str[i] := ' ';
       n := 1;
       REPEAT
         IF (ch = '_') THEN ch := ' ';
         str[n] := ch;
        IF EOLN(infile) THEN READLN(infile);
         READ(infile, ch);
         n := n + 1;
       UNTIL ((ch = ',') OR (ch = ')') OR (n > nmlngth));
       n := 1;
       REPEAT
         found := TRUE;
         FOR i := 1 TO nmlngth DO
           found := found AND (str[i] = name[n, i]);
         IF found
        THEN p := treenode[n]
         ELSE n := n + 1;
       UNTIL (n > spp) OR found;
       IF n > spp
       THEN BEGIN
         WRITE(OUTPUT, ' Cannot find species: ');
         FOR i := 1 TO nmlngth DO
           WRITE(OUTPUT, str[i]);
         WRITELN(OUTPUT);
         END;
       END;
  END; (* addelement *)

BEGIN (* treeread *)
  root := treenode[spp + 1];
  nextnode := spp;
  root^.ancestor := NIL;
  addelement (root);
  READLN(infile);
END; (* treeread *)

 PROCEDURE printree;
  (* prints out diagram of the tree *)
  CONST down = 2;
    over = 3;
   VAR tipx, tipy, i, rover : INTEGER;
```

```
  PROCEDURE coordinates (p : nodeptr);
    (* establishes coordinates of nodes *)
  BEGIN (* coordinates *)
     IF p^.tip
     THEN BEGIN
       p^.xcoord := tipx;
       p^.ycoord := tipy;
        tipx := tipx + rover;
        tipy := tipy + down;
        END
     ELSE BEGIN
      coordinates (p^.leftdesc);
      p^.xcoord := p^.leftdesc^.xcoord;
       coordinates (p^.rtdesc);
      p^.ycoord := p^.rtdesc^.ycoord;
        END;
  END; (* coordinates *)

  PROCEDURE drawline (i : INTEGER);
    (* draws one row of the tree diagram by moving up tree *)
     VAR p, q : nodeptr;
        n, j : INTEGER;
  BEGIN (* drawline *)
     p := root;
     WRITE(OUTPUT, '   ');
     REPEAT
      IF p^.leftdesc^.ycoord >= i
      THEN p := p^.leftdesc
      ELSE BEGIN
        q := p^.rtdesc;
       n := q^.xcoord - p^.xcoord;
        IF p^.ycoord = i
        THEN BEGIN
          IF rover < over
          THEN BEGIN
            IF noroot AND
             (i = (tipy-down)) AND (p^.xcoord = 0)
            THEN WRITE(OUTPUT, 'L')
            ELSE WRITE(OUTPUT, '*');
            n := n + 1;
            END
          ELSE BEGIN
            IF p^.index - spp >= 10
           THEN WRITE(OUTPUT, (p^.index - spp):2)
            ELSE WRITE(OUTPUT, (p^.index - spp):1, '-');
             END;
          FOR j := 1 TO n-2 DO WRITE(OUTPUT, '-');
          END
        ELSE BEGIN
          WRITE(OUTPUT, '!');
          FOR j := 1 TO n-1 DO WRITE(OUTPUT, ' ');
          END;
        p := q;
        END;
     UNTIL p^.tip;
     IF p^.ycoord = i
     THEN FOR j := 1 TO nmlngth DO
            WRITE(OUTPUT, name[p^.index, j]);
     WRITELN(OUTPUT);
  END; (* drawline *)

BEGIN (* printree *)
   tipx := 0;
   tipy := 1;
   rover := 100 DIV spp;
```

```
  IF rover > over THEN rover := over;
  coordinates (root);
  WRITELN(OUTPUT);
  WRITELN(OUTPUT);
  WRITELN(OUTPUT);
  FOR i := 1 TO tipy-down DO drawline (i);
   IF noroot
  THEN BEGIN
    WRITELN(OUTPUT);
    WRITE(OUTPUT, '  remember:');
    IF didreroot
    THEN WRITE(OUTPUT, ' (although rooted by outgroup)');
    WRITELN(OUTPUT, ' this is an unrooted tree!');
    END
  ELSE BEGIN
    WRITELN(OUTPUT, '  /');
    WRITELN(OUTPUT, ' /');
    END;
  WRITELN(OUTPUT);
END; (* printree *)

PROCEDURE describe;
  (* prints ancestors, steps and table of numbers of steps in
    each character *)
   VAR i, j, k : INTEGER;

   PROCEDURE hypstates;
    (* fill in and describe states at interior nodes *)
     VAR i, j, k : INTEGER;
       unknown : BOOLEAN;
       dohyp : bitarray;

     PROCEDURE filltrav (r : nodeptr);
       (* traverse to fill in interior node states *)
     BEGIN (* filltrav *)
       IF NOT r^.tip
       THEN BEGIN
         filltrav (r^.leftdesc);
         filltrav (r^.rtdesc);
          fillin (r);
          END;
     END; (* filltrav *)

     PROCEDURE hyptrav (r : nodeptr);
       (*  compute, print out states at one interior node *)
       VAR i : INTEGER;
         bottom, maybe, nonzero : BOOLEAN;
          l0, l1, r0, r1, s0, s1, a0, a1, temp, dh, wa : bitset;
         zerobelow, onebelow : bitarray;

        PROCEDURE hyprint;
          (* print out states at node *)
          VAR i, j, k  : INTEGER;
            l : bitrange;
            dot, a0, a1, s0, s1 : BOOLEAN;
        BEGIN (* hyprint *)
          IF bottom
          THEN BEGIN
            IF (noroot AND (NOT didreroot))
             THEN WRITE(OUTPUT, '        ')
             ELSE WRITE(OUTPUT, ' root   ');
             END
          ELSE WRITE(OUTPUT,
              (r^.ancestor^.index - spp):4, '    ');
          IF r^.tip
```

```
      THEN FOR i := 1 TO nmlngth DO
        WRITE(OUTPUT, name[r^.index, i])
      ELSE WRITE(OUTPUT, (r^.index - spp):4, '      ');
      IF bottom AND (noroot AND (NOT didreroot))
      THEN WRITE(OUTPUT, '          ')
      ELSE IF nonzero
        THEN WRITE(OUTPUT, '   yes    ')
        ELSE If unknown
          THEN WRITE(OUTPUT, '    ?     ')
          ELSE IF maybe
            THEN WRITE(OUTPUT, '  maybe   ')
            ELSE WRITE(OUTPUT, '   no     ');
      FOR j := 1 TO chars DO BEGIN
        newline(j, 40, nmlngth+18);
        k := (j-1) DIV bits + 1;
        l := (j-1) MOD bits + 1;
       dot := (NOT (l IN wagner[k])) AND (guess[j] = '?');
        s0 := l IN r^.statezero[k];
        s1 := l IN r^.stateone[k];
        a0 := l IN zerobelow[k];
        a1 := l IN onebelow[k];
        dot := dot OR (((NOT bottom) OR (NOT noroot)
          OR didreroot) AND ((a1 = s1) AND (a0 = s0)
          AND (NOT (s0 = s1))));
        IF dot
        THEN WRITE(OUTPUT, '.')
        ELSE BEGIN
          IF s0
          THEN WRITE(OUTPUT, '0')
          ELSE IF s1
            THEN WRITE(OUTPUT, '1')
            ELSE WRITE(OUTPUT, '?');
          END;
        IF (j MOD 5) = 0
        THEN WRITE(OUTPUT, ' ');
        END;
      WRITELN(OUTPUT);
    END; (* hyprint *)

  BEGIN (* hyptrav *)
    bottom := (r^.ancestor = NIL);
    maybe := FALSE;
    nonzero := FALSE;
    IF bottom
    THEN BEGIN
      zerobelow := zeroanc;
      onebelow := oneanc;
      END
    ELSE BEGIN
     zerobelow := r^.ancestor^.statezero;
     onebelow := r^.ancestor^.stateone;
      END;
    FOR i := 1 TO words DO BEGIN
      dh := dohyp[i];
      s0 := r^.statezero[i];
      s1 := r^.stateone[i];
      a0 := zerobelow[i];
      a1 := onebelow[i];
      IF NOT r^.tip
      THEN BEGIN
        wa := wagner[i];
       l0 := r^.leftdesc^.statezero[i];
       l1 := r^.leftdesc^.stateone[i];
       r0 := r^.rtdesc^.statezero[i];
       r1 := r^.rtdesc^.stateone[i];
```

```
            s0 := wa * (a0 * l0 + a0 * r0 + l0 * r0)
                  + dh * (fullset-wa) * s0;
            s1 := wa * (a1 * l1 + a1 * r1 + l1 * r1)
                  + dh * (fullset-wa) * s1;
            temp := fullset - (s0 + s1 + l1 + l0 + r1 + r0);
            s0 := s0 + temp * a0;
            s1 := s1 + temp * a1;
           r^.statezero[i] := s0;
           r^.stateone[i] := s1;
            END;
         maybe := maybe OR ((dh * (s0 + s1)) <> (a0 + a1));
         nonzero := nonzero OR ((s1 * a0 + s0 * a1) <> []);
          END;
       hyprint;
      IF NOT r^.tip
      THEN BEGIN
        hyptrav (r^.leftdesc);
        hyptrav (r^.rtdesc);
         END;
  END; (* hyptrav *)

BEGIN (* hypstates *)
  FOR i := 1 TO words DO BEGIN
    zeroanc[i] := [];
    oneanc[i] := [];
    END;
   unknown := FALSE;
  FOR i := 1 TO chars DO BEGIN
     j := (i-1) DIV bits + 1;
     k := (i-1) MOD bits + 1;
     IF (guess[i] = '0')
    THEN zeroanc[j] := zeroanc[j] + [k];
     IF (guess[i] = '1')
     THEN oneanc[j] := oneanc[j] + [k];
     unknown := unknown OR ((NOT (k IN wagner[j]))
                      AND (guess[i] = '?'));
     END;
  FOR i := 1 TO words DO
    dohyp[i] := wagner[i] + zeroanc[i] + oneanc[i];
  filltrav (root);
   WRITE(OUTPUT, ' From    To     Any Steps?    ');
  WRITELN(OUTPUT, 'State at upper node');
   WRITE(OUTPUT, '                              ');
  WRITELN(OUTPUT, ' ( . means same as in the node',
                  ' below it on tree)');
  WRITELN(OUTPUT);
  hyptrav (root);
END; (* hypstates *)

PROCEDURE treeout (p : nodeptr);
  (* write out file with representation of final tree *)
   VAR i, n : INTEGER;
     c : CHAR;
BEGIN (* treeout *)
  IF p^.tip
  THEN BEGIN
     n := 0;
     FOR i := 1 TO nmlngth DO
       IF name[p^.index, i] <> ' '
       THEN n := i;
     FOR i := 1 TO n DO BEGIN
       c := name[p^.index, i];
       IF c = ' '
       THEN c := '_';
       WRITE(treefile, c);
```

```
            END;
          END
        ELSE BEGIN
          WRITE(treefile, '(');
          treeout (p^.leftdesc);
          WRITE(treefile, ',');
          treeout (p^.rtdesc);
          WRITE(treefile, ')');
          END;
        IF p = root
        THEN WRITELN(treefile, ';');
      END; (* treeout *)

    BEGIN (* describe *)
      WRITELN(OUTPUT);
      WRITELN(OUTPUT, ' requires a total of ',-like:10:3);
      WRITELN(OUTPUT);
      IF weights THEN WRITE(OUTPUT, ' weighted');
      WRITELN(OUTPUT, ' steps in each character:');
      WRITE(OUTPUT, '       ');
      FOR i := 0 TO 9 DO WRITE(OUTPUT, i:4);
      WRITELN(OUTPUT);
      WRITE(OUTPUT, '     *-----------------------------------');
      WRITELN(OUTPUT, '-----');
      FOR i := 0 TO (chars DIV 10) DO BEGIN
        WRITE(OUTPUT, 10*i:6);
        WRITE(OUTPUT, '!');
        FOR j := 0 TO 9 DO BEGIN
          k := 10*i + j;
          IF (k = 0) OR (k > chars)
          THEN WRITE(OUTPUT, '    ')
          ELSE WRITE(OUTPUT, numsteps[k]+extras[k]:4);
          END;
        WRITELN(OUTPUT);
        END;
      WRITELN(OUTPUT);
      IF questions AND ((NOT noroot) OR didreroot)
      THEN BEGIN
        WRITELN(OUTPUT, ' best guesses of ancestral states:');
        WRITE(OUTPUT, '       ');
        FOR i := 0 TO 9 DO WRITE(OUTPUT, i:2);
        WRITELN(OUTPUT);
        WRITELN(OUTPUT, '     *--------------------');
        FOR i := 0 TO (chars DIV 10) DO BEGIN
          WRITE(OUTPUT, ' ');
          WRITE(OUTPUT, 10*i:5, '!');
          FOR j := 0 TO 9 DO
            IF (10*i+j = 0) OR (10*i+j > chars)
            THEN WRITE(OUTPUT, ' ')
            ELSE WRITE(OUTPUT, ' ', guess[10*i+j]);
          WRITELN(OUTPUT);
          END;
        WRITELN(OUTPUT);
        END;
      hypstates;
      WRITELN(OUTPUT);
      IF trout
      THEN treeout (root);
    END; (* describe *)

  BEGIN (* maketree *)
    fullset := [1..bits];
    IF NOT usertree
    THEN BEGIN
      FOR i := 1 TO spp DO enterorder[i] := i;
```

```
 IF jumble
 THEN FOR i := 1 TO spp DO BEGIN
   j := TRUNC(randum(seed)*spp) + 1;
   k := enterorder[j];
  enterorder[j] := enterorder[i];
   enterorder[i] := k;
    END;
add (treenode[enterorder[2]],
  treenode[enterorder[1]], treenode[spp+1]);
root := treenode[spp+1];
WRITELN(OUTPUT, ' Adding species:');
 WRITE(OUTPUT, '    ');
 FOR i := 1 TO nmlngth DO
  WRITE(OUTPUT, name[enterorder[1],i]);
WRITELN(OUTPUT);
 WRITE(OUTPUT, '    ');
 FOR i := 1 TO nmlngth DO
  WRITE(OUTPUT, name[enterorder[2],i]);
WRITELN(OUTPUT);
lastrearr := FALSE;
FOR i := 3 TO spp DO BEGIN
   bestyet := -32000;
   there := root;
  item := treenode[enterorder[i]];
   nufork := treenode[spp + i - 1];
  addpreorder (root, item, nufork);
   add (there, item, nufork);
  IF root^.ancestor <> NIL THEN root := root^.ancestor;
   like := bestyet;
  rearrange (root);
   WRITE(OUTPUT, '    ');
   FOR j := 1 TO nmlngth DO
    WRITE(OUTPUT, name[enterorder[i],j]);
  WRITELN(OUTPUT);
  lastrearr := (i = spp);
  IF lastrearr
  THEN BEGIN
    WRITELN(OUTPUT, ' Doing global rearrangements');
    nextree := 1;
    bestlike := bestyet;
    REPEAT
      WRITE(OUTPUT, '    ');
      gotlike := bestlike;
      FOR j := 1 TO nonodes DO BEGIN
        bestyet := -32000;
        there := root;
       item := treenode[j];
        IF item <> root
        THEN BEGIN
         nufork := treenode[j]^.ancestor;
          IF root = nufork
          THEN BEGIN
            IF nufork^.leftdesc = item
            THEN root := nufork^.rtdesc
            ELSE root := nufork^.leftdesc;
             END;
          remove (item, nufork);
         addpreorder (root, item, nufork);
          add (there, item, nufork);
          IF root^.ancestor <> NIL
          THEN root := root^.ancestor;
           END;
        WRITE(OUTPUT, '.');
        END;
      WRITELN(OUTPUT);
```

```
      UNTIL bestlike <= gotlike;
      END;
    END;
  FOR i := spp DOWNTO 2 DO
   remove (treenode[i], dummy);
  WRITELN(OUTPUT);
  IF (nextree = 2)
  THEN WRITELN(OUTPUT, ' One most parsimonious tree found:')
  ELSE WRITELN(OUTPUT, nextree-1:6, ' trees in all found');
  IF nextree > maxtrees + 1
  THEN BEGIN
    WRITELN(OUTPUT, ' here are the first', maxtrees:4,
              ' of them');
    nextree := maxtrees + 1;
    END;
  WRITELN(OUTPUT);
  FOR i := 1 TO nextree-1 DO BEGIN
   add (treenode[2], treenode[1], treenode[spp+1]);
    root := treenode[spp+1];
    FOR j := 3 TO spp DO BEGIN
     add(treenode[bestrees[i][j]], treenode[j],
                                  treenode[spp+j-1]);
      IF root^.ancestor <> NIL THEN root := root^.ancestor;
      END;
    IF noroot
   THEN reroot (treenode[outgrno]);
   didreroot := outgropt AND noroot;
   evaluate (root);
   printree;
   describe;
   FOR j := 2 TO spp DO
     remove (treenode[j], dummy);
   END;
  IF trout
  THEN BEGIN
    WRITELN(OUTPUT, ' Trees also written onto file');
    WRITELN(OUTPUT);
    END;
  END
ELSE BEGIN
  READLN(infile, numtrees);
  IF numtrees > maxuser
  THEN BEGIN
    WRITE(OUTPUT, ' TOO MANY USER-DEFINED TREES');
    WRITELN(OUTPUT, '  only the first', maxuser:4, ' used');
    numtrees := maxuser;
    END;
  WRITE(OUTPUT, ' User-defined tree');
  IF numtrees > 1
  THEN WRITE(OUTPUT, 's');
  WRITELN(OUTPUT, ':');
  FOR which := 1 TO numtrees DO BEGIN
    treeread;
    didreroot := outgropt AND noroot;
    IF noroot
   THEN reroot (treenode[outgrno]);
    evaluate (root);
    printree;
    describe;
    END;
  WRITELN(OUTPUT);
  WRITELN(OUTPUT);
  IF (numtrees > 1) AND (chars > 1)
  THEN BEGIN
    WRITE(OUTPUT, ' Tree    Steps     Diff Steps    Its S.D.');
```

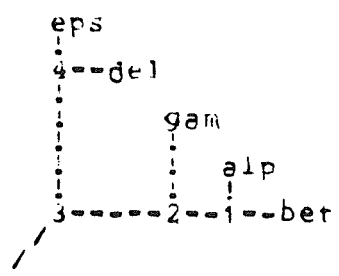
```
          WRITELN(OUTPUT, '  Significantly worse?');
          WRITELN(OUTPUT);
          FOR which := 1 TO numtrees DO BEGIN
            WRITE(OUTPUT, which:4, nsteps[which]:10:3);
            IF which = minwhich
            THEN WRITELN(OUTPUT, ' <------- best')
            ELSE BEGIN
               sum := 0.0;
               sum2 := 0.0;
              FOR j := 1 TO chars DO BEGIN
                sum := sum + fsteps[which][j]-fsteps[minwhich][j];
                 sum2 := sum2
                     + SQR(fsteps[which][j]-fsteps[minwhich][j]);
                END;
             sd := SQRT((chars/(chars-1))*(sum2-SQR(sum/chars)));
              WRITE(OUTPUT, (nsteps[which]-minsteps):13:5, sd:13:5);
              IF sum > 1.95996*sd
              THEN WRITELN(OUTPUT, '          Yes')
              ELSE WRITELN(OUTPUT, '          No');
              END;
            END;
          WRITELN(OUTPUT);
          WRITELN(OUTPUT);
          END;
        END;
  END; (* maketree *)

BEGIN (* Mixed parsimony by uphill search *)
  (* reads in spp, chars, and the data. Then calls maketree to
    construct the tree *)
  (* ASSIGN(infile, 'INFILE');  Turbo Pascal only *)
  RESET(infile);
  doinput;
  IF trout
  THEN BEGIN
    (* ASSIGN(treefile, 'TREEFILE');  Turbo Pascal only *)
    REWRITE(treefile);
    END;
  maketree;
(* CLOSE(infile);  -- may need this here *)
  IF trout
  THEN BEGIN
  (* CLOSE(treefile);  -- may need this here *)
    END;
END. (* Mixed parsimony by uphill search *)
```

ANNEXE IV : Exemples supplémentaires d'"output" de MIX.

```
Mixed parsimony algorithm, version 3.1

 3 species,   7  characters

Wagner parsimony method

Ancestral states:
              00001 10


Name          Characters
----          ----------

un            00000 00
de          ►P1111 10
tr          ►P1111 11


Adding species:
   un
   de
   tr
Doing global rearrangements
   .....

One most parsimonious tree found:



  un
  !
  !  tr
  !  !
  1--2--de
 /
/


requires a total of       8.000

steps in each character:
         0   1   2   3   4   5   6   7   8   9
     *-----------------------------------------
    0!       2   1   1   1   1   1   1

From     To      Any Steps?     State at upper node
                                ( . means same as in the node below it on tree)

root       1          no       ..... ..
  1     un            yes      ....0 0.
  1        2          yes      .111. ..
  2     tr            yes      ?.... .1
  2     ce           maybe     ?.... ..
```

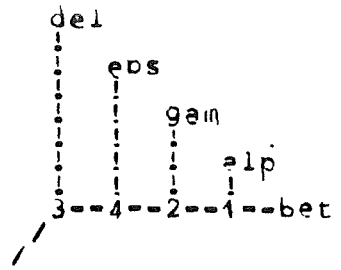
Fig. IV.1 Utilisation par MIX de l'état "P" (polymorphe).

```
Mixed parsimony algorithm, version 3.1
 5 species.   6  characters

Parsimony methods:
                SWWWW W

Ancestral states:
                0???? ?

Name            Characters
----            ----------

alp             11011 0
bet             11000 0
gam             10011 0
del             00100 1
eps             00111 0

Adding species:
   alp
   bet
   gam
   del
   eps
Doing global rearrangements
   .........

    3 trees in all found
```




```
requires a total of       8.000

steps in each character:
         0   1   2   3   4   5   6   7   8   9
      *-------------------------------------------
    0!       1   1   1   2   2   1

best guesses of ancestral states:
       0 1 2 3 4 5 6 7 8 9
      *-------------------

    0!   0 0 ? 1 1 0

From     To      Any Steps?     State at upper node
                                ( . means same as in the node below it on tree)

root       3         no       ..?.. .
  3        4         maybe    ..1.. .
  4     eps          no       ..... .
  4     del          yes      ...00 1
  3       2          yes      1.0.. .
  2     gam          no       ..... .
  2         1        yes      .1... .
  1     alp          no       ..... .
  1     bet          yes      ...00 .
```

Fig.IV.2 : Illustration du choix des options A & M, et de l'utilisation par MIX de l'état "?". (voir suite p. IV.3)



requires a total of 8.000

steps in each character:

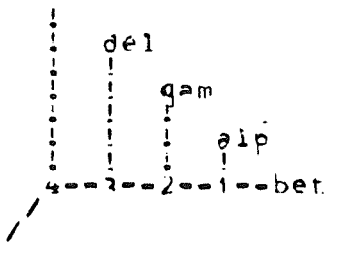
	0	1	2	3	4	5	6	7	8	9
0!		1	1	1	2	2	1			

best guesses of ancestral states:

	0	1	2	3	4	5	6	7	8	9
0!	0	0	1	?	?	?				

From	To	Any Steps?	State at upper node (. means same as in the node below it on tree)
root	3	no	...?? ?
3	del	maybe	...00 1
3	4	maybe	...11 0
4	eps	no	..... .
4	2	yes	1.0.. .
2	gam	no	..... .
2	1	yes	.1... .
1	alp	no	..... .
1	bet	yes	...00 .

eps



requires a total of 8.000

steps in each character:

	0	1	2	3	4	5	6	7	8	9
0!		1	1	1	2	2	1			

best guesses of ancestral states:

	0	1	2	3	4	5	6	7	8	9
0!	0	0	1	1	1	0				

From	To	Any Steps?	State at upper node (. means same as in the node below it on tree)
root	4	no	..... .
4	eps	no	..... .
4	3	no	..... .
3	del	yes	...00 1
3	2	yes	1.0.. .
2	gam	no	..... .
2	1	yes	.1... .
1	alp	no	..... .
1	bet	yes	...00 .

Fig. IV.2 (suite).

ANNEXE V : Exemple supplémentaire d'"output" de PAUP.

```
Option settings:

     NOTU ....................        5
     NCHAR ...................        6
     User-tree(s) ............       NO
     HYPANC ..................        1
     ADDSEQ ..................    ASIS
     HOLD ....................        1
     SWAP ....................       NO
     MULPARS .................       NO
     OPT .....................   FARRIS
     ROOT ..................... OUTGROUP
     Weights applied .........       NO
     OUTWIDTH ................       80
     Missing data code .......     None
     MAXTREE .................      N/A


Branch lengths and linkages for unrooted tree no.  1

                          Connected          Branch
      Node                to node            length

     beta (  2)*             6               1.000
    gamma (  3)              7               0.000
    delta (  4)              6               3.000
    epsil (  5)              7               2.000
        6                    8               2.000
        7                    8               0.000
        8                  alp (  1)*        1.000

► * Designated outgroup taxa


Statistics for tree no.  1

     Length =     9.000
     Consistency index = 0.667

► Warning: Rooting such that specified ingroup is monophyletic
          not possible.  Tree will be rooted using midpoint method.
```

Fig. V.1 : Un fragment d'"output" de PAUP illustrant l'impossibilité d'enracinement secondaire pour un outgroup donné (composé de plusieurs taxons).